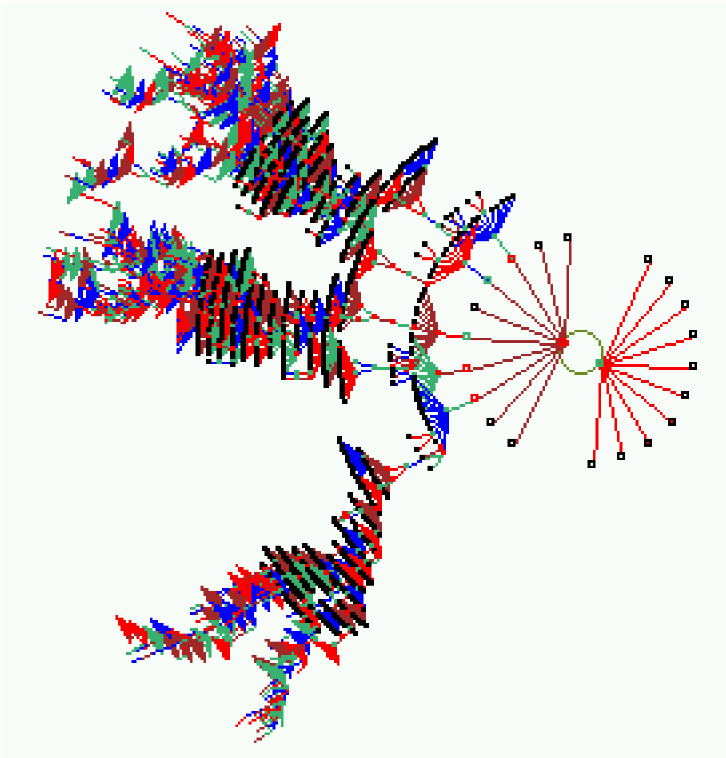


The DDLab Manual

Discrete Dynamics Lab

Tools for researching discrete dynamical networks,
from Cellular Automata to Random Boolean Networks and beyond

www.santafe.edu/~wuensch/ddlab.html

July 2001

Andrew Wuensche
andy@ddlab.com

Discrete Dynamics Inc
www.ddlab.com
7 Calle Andreita, Santa Fe, NM 87506, USA

Preface

Networks of sparsely inter-connected elements having discrete values and updating in discrete time (in parallel or sequentially) are central to a wide range of natural and artificial phenomena drawn from many areas of science; from physics to biology to cognition; to social and economic organization; to parallel computation and artificial life; to complex systems in general.

Such "decision making" networks are increasingly applied as idealized models, especially in the study of complexity and emergence, and in the behavior of networks in general, including neural and gene networks. In addition, the networks themselves have intrinsic interest as mathematical/physical systems with a large body of literature devoted to their study. Because the dynamics is difficult to describe by classical mathematics, computer simulation is required, and there is a need for simulation software for non-experts in programming to model networks in their particular fields.

DDLab is an interactive graphics program aimed at addressing these issues, widely used in both research and education. The dynamics of a wide range of finite binary networks may be investigated, from Cellular Automata to Random Boolean Networks and beyond.

As well as generating space-time patterns in one, two or three dimensions, DDLab constructs attractor basins, graphs that link network states according to their transitions, representing "global" dynamics, analogous to the phase portrait in continuous dynamical systems. The graphs, consisting of trees rooted on attractor cycles, reveal the network's "memory", how the network hierarchically categorizes state space. Learning and "inverse problem" algorithms are included that seek to reconstruct a network that satisfies a given set of transitions.

DDLab is an applications program, it does not require writing code. Network parameters and the graphics presentation can be flexibly set, reviewed and altered, including changes "on the fly". A wide variety of measures, data, analysis and statistics are available. This manual provides a comprehensive guide.

Acknowledgments

Many people have influenced DDLab by contributing ideas, suggesting new features, providing encouragement, criticism, and helping with programming. I reserve all the blame for its shortcomings. I would like to thank Mike Lesser, Grant Warrel, Crayton Walker, Chris Langton, Stuart Kauffman, Wentian Li, Pedro de Oliviera, Inman Harvey, Phil Husbands, Guillaume Barreau, Josh Smith, Raja Das, Christian Reidys, Brosl Hasslacher, Steve Harris, Simon Frazer, Burt Voorhees, John Myers, Roland Somogyi, Andrew Adamatzky, Mark Tilden, Rodney Douglas, Terry Bossomaier, Ed Coxon, Oskar Itzinger, Pietro diFenizio, Pau Fernandez, Ricard Sole, and many other friends and colleagues (to whom I apologize for not listing). Also DDLab users, and researchers and staff at the Santa Fe Institute since the spring of 1990, when I first turned up there summoned by Stuart Kauffman.

Contents

List of Figures

List of Tables

Chapter 1

Overview

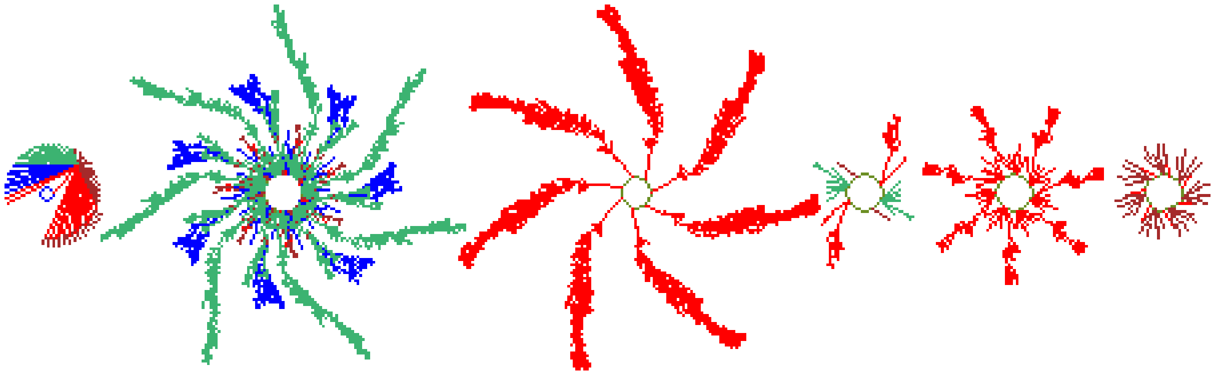


Figure 1.1: The basin of attraction field of a Cellular Automaton, $k = 3$, $n = 14$, equivalent basins suppressed.

1.1 Introduction

DDLab is an interactive graphics program for researching the dynamics of finite binary networks, from Cellular Automata to random Boolean networks and beyond, including their attractor basins. The program is relevant to the study of complexity, emergent phenomena, neural computation and aspects of theoretical biology such as modeling gene regulatory networks, and to the study of discrete dynamical networks in general in a variety of fields.

Networks can be created with any architecture from cellular automata to random Boolean networks. Cellular automata have a homogeneous rule and a nearest neighbor connections, whereas random Boolean networks have arbitrary connections, and rules which may be different at each site. The results in references [12]-[19] in the bibliography may be implemented with DDLab.

1.2 Source code and platforms

The DDLab source code is written in C. The program is maintained for the following platforms: DOS/PC, UNIX/XWindows-Sun OS 4.1 and Solaris, Linux, and Irix SGI. DDLab has been ported to hpux/X11/HP by users (see section 2.16). This manual covers all versions, which function in essentially the same way, though aspects of the graphics presentation might be slightly different.

1.3 Discrete dynamical networks

A discrete dynamical network in DDLab can be imagined as a software simulation of a collection light bulbs which transmit information to each other about their on/off state, and turn on or off according to the arriving signals. In more abstract terminology, the network is made up of elements[1] or "cells", connected to each other by directed links or "wires", where a wire has an input and output terminal. A cell takes on a value of either 0 or 1, and transmits this value down its output wires. Its value is updated as a function of the values on its input wires. Updating is usually done in parallel, in discrete "time-steps", but may also be sequential in a predetermined order.

This is the system in a nutshell. It remains to set up the network according to the various parameters,

- The number of elements, the system size, n.
- How the elements are arranged in space: a 1d, 2d or 3d array with axial dimensions i, j, h, or some other arrangement. This network "geometry" may have real meaning (depending on the "wiring scheme" below), or it may simply allow convenient indexing and representation.
- The number of input wires, k, to each cell, or the "k-mix" if k is not homogeneous.
- The "wiring scheme". Defining the location of the output terminals of each cell's input wires, the element's "neighborhood". Cellular automata have a homogeneous "nearest neighbor" (local) neighborhood throughout the network. Random Boolean networks may have a completely arbitrary wiring scheme (a "pseudo-neighborhood"), assigned at random, or the wiring scheme may be biased in some way, for example, by confining an element's pseudo-neighborhood close to itself. The wiring scheme also defines boundary conditions. Cellular automata wiring requires "periodic boundary conditions", where an array's edges wrap around to their opposite edges.
- The "rule scheme", the rules or Boolean functions in the network. Each element applies a rule to its inputs to compute its output. Usually this is made into a look-up table, the "rule-table", listing the outputs of all possible input patterns. Cellular automata have a homogeneous rule scheme, the same rule throughout the network. Random Boolean networks may have a completely arbitrary rule scheme, or again, it may be biased in some way.

[1] The term "cell" (as in cellular automata) denotes a network "element", and should not be confused with a biological cell. In random Boolean network models of gene regulatory networks a network element represents a gene, whereas in neural network models a network element represents a neuron.

DDlab is able to create these networks, and graphically represent and analyze both the networks themselves and the dynamics resulting from the changing patterns of 0s and 1s as the complex feedback web unfolds.

The networks, whether cellular automata or random Boolean networks, may be set up in one, two or three dimensions, with periodic boundary conditions. The neighborhood (or "pseudo-neighborhood") size, k, may vary from 0 to 13, and the network may have a mix of k sizes. Network wiring can also be set up to correspond to a regular n-dimensional hypercube, where $k = log_2(n)$. Network updating may be sequential as well as parallel, noisy as well as deterministic.

"Cellular automata" is henceforth abbreviated to "CA", and "random Boolean networks" to "RBN".

1.4 Space-time patterns and attractor basins

DDLab has two alternative ways of looking at network dynamics. *Local* dynamics, running the network forwards, and *global* dynamics, which entails running the network backwards.

Running forwards (local dynamics) generates the network's space-time patterns from a given initial state. Many alternative graphical representations (and methods for gathering/analyzing data) of space-time patterns are available to illustrate different aspects of local network dynamics, including "filtering" to show up emergent structures more clearly.

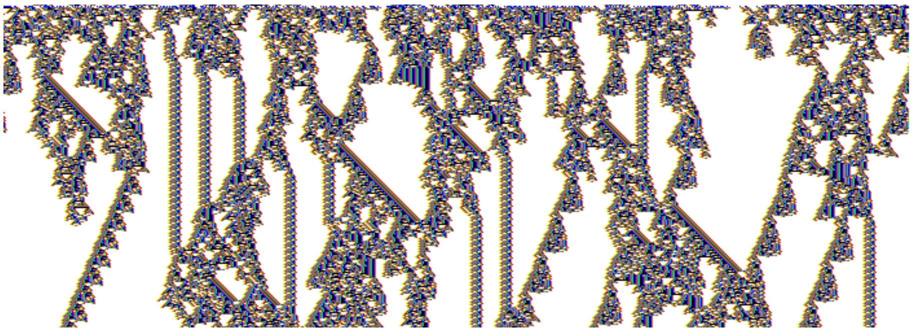


Figure 1.2: The space-time pattern of a 1d complex CA with interacting gliders. 308 time-steps from a random initial state. System size n=700, Neighborhood size k=7, rule (hex) = 3b 46 9c 0e e4 f7 fa 96 f9 3b 4d 32 b0 9e d0 e0. Cells are colored/shaded according to neighborhood look-up instead of the value. Space is across and time down the page. The basin of attraction field for this rule for n=16 is shown figure 1.3.

Running "backwards" (global dynamics) generates multiple predecessors rather than a trajectory of unique successors. This procedure reconstructs the branching sub-tree of ancestor patterns rooted on a particular state. States without predecessors may be disclosed, the so called "garden-of-Eden" states, the "leaves" of the sub-tree. Sub-trees, basins of attraction (with a topology of trees rooted on attractor cycles), or the entire basin of attraction field (referred to collectively as "attractor basins") can be displayed as directed graphs in real time, with many presentation options, and methods for gathering/analyzing data. The attractor basins of "random maps" may be generated, with or without some bias in the mapping.

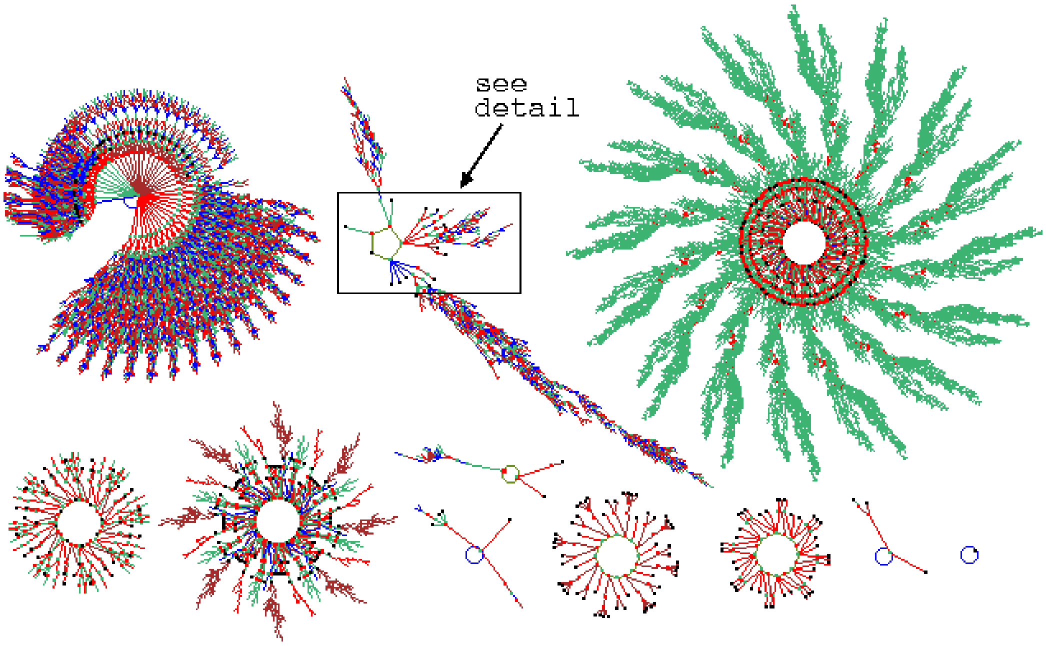


Figure 1.3: The basin of attraction field of a complex CA rule. An example of its space-time patterns are shown in figure 1.2. n=16, k=7. The 2^{16} = 65536 states in state space are connected into 89 basins of attraction. The 11 non-equivalent basins are shown, with symmetries characteristic of CA[12]. The period (p), percentage of state space in each basin type(s), and number of each type (t), of the biggest three basins (top row), are as follows: (1) p=1 s=15.7% t=1. (2) p=5 s=55.8% t=16. (3) p=192 s=22.9% t=1. The field's G-density=0.451, λ_{ratio}=0.938, Z=0.578.

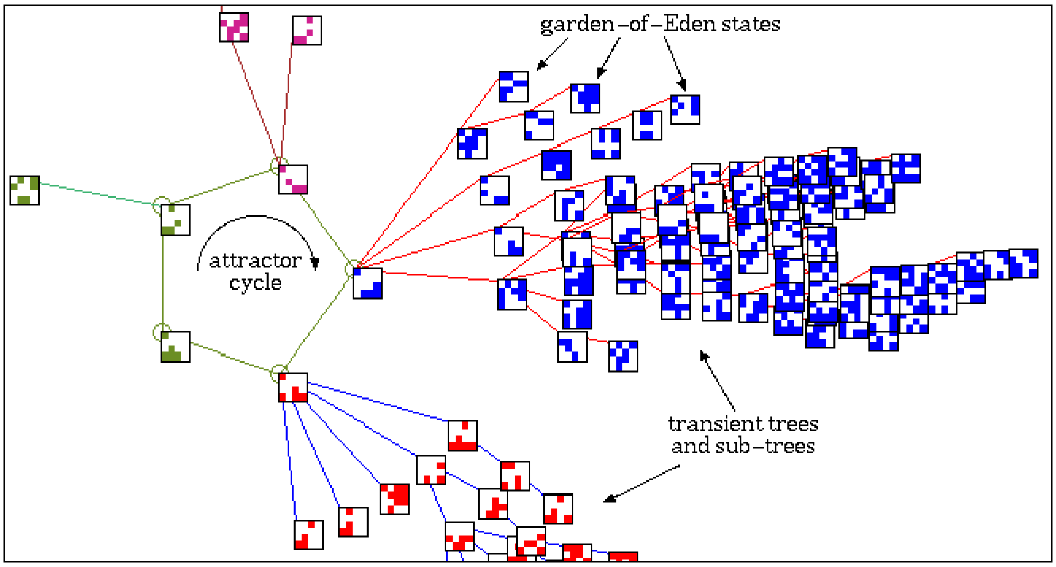


Figure 1.4: A detail of the 2nd basin of attraction in figure 1.3. The states are shown as 4×4 bit patterns.

1.5 Categorization

It can be argued that attractor basins represent the network's "memory" by their hierarchical categorization of state-space. Each basin is categorized by its attractor and each sub-tree by its root. Learning/forgetting algorithms allow attaching/detaching sets of states as predecessors of a given state by automatically mutating rules or changing connections. This allows sculpting the basin of attraction field to approach a desired scheme of hierarchical categorization. More generally, preliminary "inverse problem" or "reverse engineering" algorithms are included to find the network that satisfies a given set of transitions.

1.6 Network size limitations

Whereas large networks may be run forward to look at space-time patterns, or backward to look at subtrees, the size is limited when generating the entire basin of attraction field, given that state-space grows exponentially with system size n. The program's upper limit for basin of attraction fields is n=31 (lower in practice). Otherwise the limit is n=65025, based on the maximum size of a square 2d network, 255×255, where n can be represented by an unsigned short int. This applies to space-time patterns (in 1d, 2d and 3d), and to single basins or sub-trees, though in practice much smaller sizes are appropriate for attractor basins. Larger sizes may be tried, but may impose unacceptable time, memory or display constraints.

Networks with disordered or "chaotic" dynamics, with low in-degree or branchiness in their subtrees, allow backwards computation of much larger networks than for ordered dynamics which have high in-degree. For CA, rules giving chaotic dynamics have a high Z parameter, rules giving ordered dynamics have a low Z parameter[19].

For RBN, running backwards generally imposes a greater computational load than for CA.

1.7 Parameters and options

DDLab is an applications program, it does not require writing code. The network's parameters, and the graphics display and presentation, can be very flexibly set, reviewed and altered from DDLab's graphical user interface.

Changes can be made on-the-fly, including changes to rules, connections, current state, scale, and alternative presentations highlighting different properties of space-time patterns. Networks of whatever dimension can be interchangeably represented in 1d, 2d, and 3d, as well as in a 3d isometric projection. The latter is especially useful for showing the space-time patterns of 2d networks such as Conway's "game-of-Life" [2].

The network architecture, states, data, and the screen image can be saved and loaded in a variety of tailor-made file formats.

1.8 Measures and data

Various quantitative, statistical and analytical measures and data on both forward dynamics and attractor basin topology are available in DDLab, as well as various global parameters for rules and network architecture. The measures and data, shown graphically as well as numerically, include the following:

- Rule parameters λ (or P) and Z.
- The frequency of canalizing inputs. This can be set to any arbitrary level.
- Various measures on forward dynamics such as pattern density, frozen islands, damage spread between two networks, the Derrida plot, rule-table lookup frequency (which allows "filtering"), input entropy, and the variance of the input entropy, which allows ordered, complex and chaotic rules to be discriminated automatically.
- Various global measures on the topology of attractor basins including garden-of-Eden density and a histogram of in-degree frequency.

1.9 Contents summary

- Chapter 2 provides a descriptive summary of DDLab's functions.
- Chapter 3 describes how DDlab can be accessed.
- Chapter 4 describes DDLab's graphical user interface and gives some "quick start" examples. Its probably a good idea to try these right away to get the flavor of the DDLab before reading on, or tackling the detailed reference manual.
- Chapters 5 to 35 contain the detailed reference manual.

For further background on the attractor basins of CA and random Boolean networks, and their implications, see references [12]-[19] in the bibliography.

DDLab is in the process of continual development. New features are being added in response to various research needs. Some aspects of this manual may be out of date or not applicable to particular platforms.

For further information about DDLab, including downloading, documentation, licensing, platforms, updates, examples of output, applications, related publications, links, mailing list etc., refer to the DDLab web page at one of the following sites,

- `www.ddlab.com`
- `www.santafe.edu/~wuensch/ddlab.html`
- `www.cogs.susx.ac.uk/users/andywu/ddlab.html`

Chapter 2

Summary of DDLab functions

This chapter provides a descriptive summary of DDLab's functions.

2.1 DDLab's graphical user interface

DDLab's graphical user interface (see chapter 5) allows setting, viewing and amending network parameters by responding to prompts or accepting defaults. The prompts present themselves in a main sequence and also in a number of context dependent prompt windows. You can backtrack to previous prompts in the sequence, and in some cases skip forward. A flashing cursor indicates the current prompt. "Return" (or the left mouse button) steps forward through the prompts, "q" (or the right mouse button) backtracks, or interrupts a run.

2.2 The neighborhood k or k-mix

The size of the "neighborhood", the number of inputs each cell receives, k, can vary from 0 to 13 (see chapter 8) . k can be homogenious, or there can be a mix of k-values in the network. The k-mix may be set and modified in a variaty of ways, including defining the proportions of different k's to be allocated at random in the network. A k-mix may be saved/loaded from a file, but is also implicit in the wiring scheme (see below).

2.3 Wiring

The network's wiring scheme (i.e. its connections) has default settings for regular CA (for 1d, 2d and 3d) for each neighborhood size, k=0 to 13 (see chapter 9). Wiring can also be set at random (non-local wiring), with a wide variety of constraints and biases, or by hand (see chapter 11). A wiring scheme can be set and amended just for a predefined sub-network within the network, and may be saved/loaded from a file (see chapter 19).

In most cases regular 2d wiring defines a square grid on the torus, and includes the von Neumann and Moore neighborhoods of 5 and 9, cells. However the 6 and 7 cell regular 2d neighborhood is wired to define a triangular grid. Regular 3d wiring defines a cubic grid with periodic boundary conditions.

Non-local wiring can be constrained in various ways, including confinement within a local patch of cells with a set diameter in 1d, 2d and 3d. Part of the network only can be designated to accept a particular type of wiring scheme, for example rows in 2d and layers in 3d. The wiring can be biased to connect designated rows or layers.

The network parameters can be displayed and amended in a 1d, 2d or 3d graphic format, in a "spread sheet" (see chapter 17), or as a network graph which can be rearranged in various ways, including dragging nodes with the mouse (see chapter 20).

2.4 Rules

A network may have one homogenious rule as for CA, or a rule-mix as for RBN. The rule-mix can be confined to a subset of (selected) rules. Rules (and totalistic codes) may be set and modified in a wide variety of ways, in decimal, hex, as a rule-table bit pattern (using the mouse), at random or loaded from a file. The "game-of-Life", "majority", and other predefined rules or rule biases can be selected (see chapters 12, 14, 16).

A rule scheme can be set and amended just for a predefined sub-network within the network, and may be saved/loaded from a file (see chapter 19).

Rules may be changed into their equivalents (by reflection and negative transformations), and transformed into equivalent rules with larger or smaller neighborhoods (see chapter 18). Rules transformed to larger neighborhoods are useful to achieve finer mutations. Rule parameters λ and Z, and the frequency of canalizing inputs in a network can be set to any arbitrary level (see chapter 15).

2.5 Initial network state, seed

An initial network state (the seed) is required to run the network forward and generate space-time patterns. A seed is also required to run backwards to generate a sub-tree or single basin. A basin of attraction field does not require a seed.

As in setting a rule, there are a wide variaty of methods for defining the seed (see chapter 21), in decimal or hex, as a bit pattern in 1d, 2d or 3d, at random (with various constraints or biases), or loaded from a file (see chaper 35). The bit pattern method is a mini drawing program, using the mouse (or keyboard) to set cell values (0,1), particularly useful for 2d and 3d.

2.6 Networks of sub-networks

Its possible to create a system of independent or weakly coupled sub-networks within the base network, either directly, or by saving smaller networks to a file, then loading them at appropriate positions in the base network (see chapter 19). Thus a 2d network can be tiled with sub-networks, and 1d, 2d or 3d sub-networks can be inserted into a 3d base network.

The parameters of the sub-networks can be totally different from the base network, provided the base network is set up appropriately, with sufficent "memory" to accomadate the sub-network. For example, to load an RBN into a CA, the CA may need be set up as if it were an RBN. To load a mixed-k sub-network into single-k base network, k in the base network needs to be at least as big as the biggest k in the sub-network. Options are available to easily set up networks in this way. Once loaded, the wiring can be fine-tuned to interconnect the sub-networks.

A network can be automatically duplicated to create a total network made up of two identical sub-networks. This is useful to see the difference pattern (or damage spread) between two networks from similar initial states (see section 31.15).

2.7 Presentation options

Many options are provided for the presentation of attractor basins and space-time patterns. Again, many of these settings can be changed "on the fly".

2.7.1 Space-time patterns

(see chapters 31 and 32)

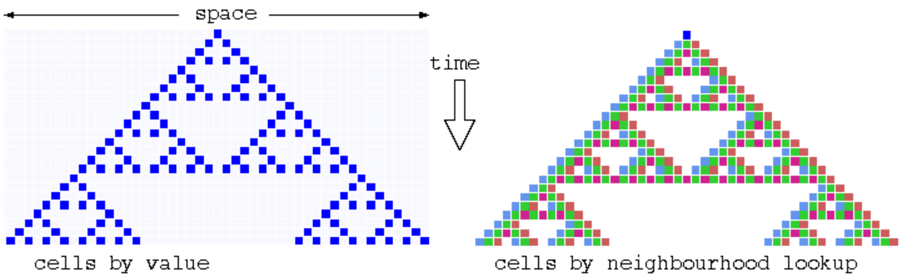


Figure 2.1: Space-time patterns of a 1d CA (n=24, k = 3, rule 90). 24 time-steps from an initial state with a single central 1. Two alternative presentations are shown. *Left*, cells by value, light=0 dark=1. *Right*, cells colored according to their look-up neighbourhood.

Cells in space-time patterns are coloured according to their value (0,1) or alternatively according to their neighborhood at the previous time step, the entry in the look-up table that determined the cells' value. A key press will toggle between the two. Space-time patterns can be filtered to suppress cells that updated according to the most frequently occuring neighborhoods, thus exposing "gliders" and other structures (see section 32.10.5).

The presentation can be set to highlight cells that have not changed in the previous x generations, where x can be set to any value (see section 32.10.1). The emergence of such frozen elements is associated with "canalizing inputs", and underlies Kauffman's RBN model of gene regulatory networks[7, 4].

A 1d space-time pattern may be presented in successive vertical sweeps, or may be continuously scrolled. 2d networks such as the "game-of-Life" can be displayed simply on a 2d grid, and also as a space-time pattern (2d+time) in a 3d isometric projection. 3d networks are presented within a 3d "cage". The presentation of space-time patterns can be switched "on the fly" between 1d, 2d, 2d+time, and 3d, irrespective of their native dimensions. DDLab automatical unravels or bundles up the dimensions. There are many other on-the-fly options, including changing the scale of space-time patterns, changing the seed, rule/s, wiring, and the size of 1d networks (see chapter 32).

Concurrently with these standard presentations, space-time patterns can be displayed in a separate window according to the network graph layout. This can be rearranged in many ways, including various default layouts. For example a 1d space-time pattern to be shown in a circular layout (see section 32.15).

2.7.2 Attractor basins

(see chapters 24 to 30)

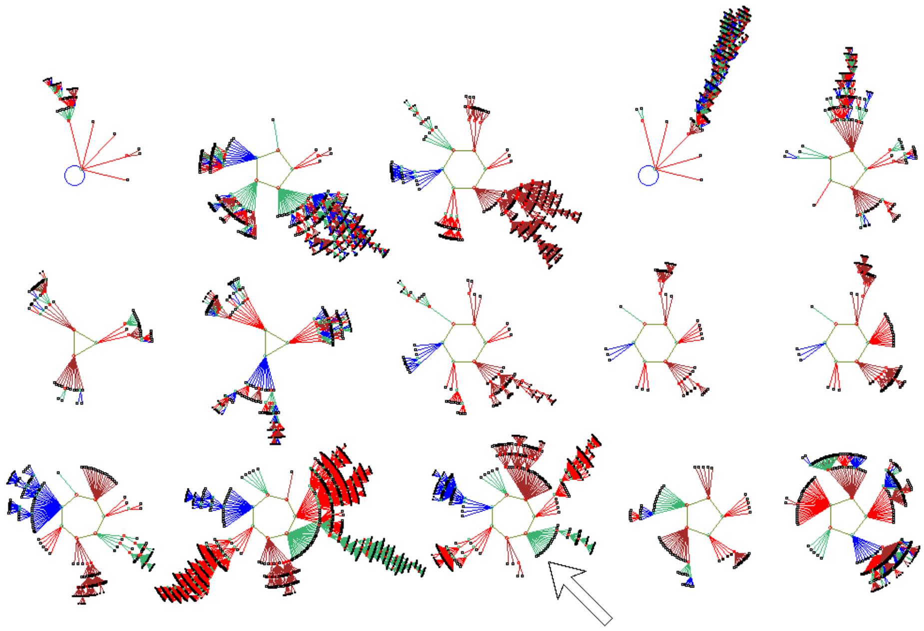


Figure 2.2: The basin of attraction field of a random Boolean network (n=13, k=3). The $2^{13} = 8192$ states in state space are organized into 15 basins, with attractor periods ranging between 1 and 7. The number of states in each basin is: 68, 984, 784, 1300, 264, 76, 316, 120, 64, 120, 256, 2724, 604, 84, 428. Figure 2.3 shows the arrowed basin in more detail. Right: the network's architecture, its wiring/rule scheme.

cell	wiring	rule
12	10,1,7	86
11	6,2,9	4
10	10,10,12	196
9	2,10,4	52
8	5,6,8	234
7	12,5,12	100
6	1,9,0	6
5	5,7,5	100
4	4,11,7	6
3	8,12,12	94
2	11,6,12	74
1	6,5,9	214
0	12,9,6	188

Options for attractor basins allow the selection of the basin of attraction field, a single basin (from a selected seed), or a sub-tree (also from a seed). Because a random seed is likely to be a garden-of-Eden state, to generate sub-trees an option is offered to run the network forward a given number of steps to a new seed before running backward. This guarantees a sub-tree with at least that number of levels.

Options (and defaults) are provided for the layout of attractor basins, their size, position, spacing, and type of node display (as a spot, in decimal, hex or a 1d or 2d bit pattern, or none). Regular 1d and 2d CA produce attractor basins where sub-trees and basins are equivalent by rotational symmetry. This allows "compression" of basins (by default) into non-equivalent prototypes, though compression can be turned off. Attractor basins are generated for a given system size, or for a range of sizes. As attractor basins are generating, the reverse space-time pattern can be simultaniously displayed.

An attractor basin run can be set to pause to see data on each transient tree, each basin, or each field. Any combination of this data, including the complete list of states in basins and trees, can be saved to a file (see chapter 27).

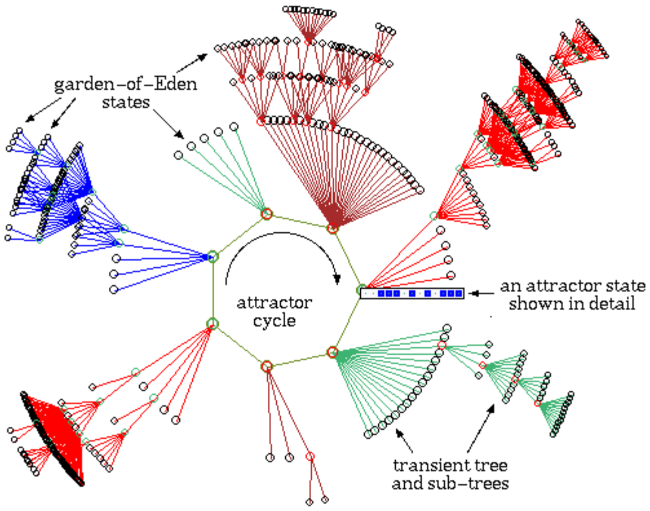


Figure 2.3: A basin of attraction (one of 15) of a random Boolean network (n=13, k=3) shown in figure 2.2. The basin links 604 states, of which 523 are garden-of-Eden states. The attractor period = 7, and one of the attractor states is shown in detail as a bit pattern. The direction of time is inwards from garden-of-Eden states to the attractor, then clock-wise.

Normally a run will pause before the next "mutant" attractor basin, but this pause may be turned off to create a continuous demo of new attractor basins. A "screensave" demo option shows new basins continually growing at random positions (see section 24.4).

2.7.3 Interrupting a run

At any time, a space-time pattern or attractor basin run can be interrupted to pause, save or print the screen image, change various parameters, or backtrack through options (see chapters 32 and 30).

2.8 Graphics

(see chapter 5)

In DOS the graphics will start up at a resolution of 640×480 (VGA) but can be started at a higher resolution with a program parameter, or the resolution can be changed later.

In the Unix/Linux version the DDLab window starts up at 1024×768 or a size that is automaticaly set to comfortably fit on the monitor, and can be resized, moved and iconised in the usual way.

The default background color is black, but can be reset to white either from within the program or with a program parameter.

The text size and spacing is set automatically according to the screen resolution, but can be resized.

2.9 Filing and Printing

DDLab allows filing a wide range of filetypes, including network parameters, data, and the screen image (see sections 19 and 35). For compatability with DOS, filenames follow the DOS format, so a file name has up to eight characters (starting with a letter) plus a 3 character extension. For example `myfile25.dat`. In DDLab, only the first part of the filename is selected without the extention (or the default filename provided is accepted), the extension is set automatically to identify the file type.

2.9.1 Filing network parameters

indexfiling!network parameters Network parameters and states can be saved and loaded for the following: k-mix, wiring-schemes, rules, rule-schemes, wiring/rule schemes, and network states (see chapter 19).

2.9.2 Filing data

Data on attractor basins, at various levels of detail (see chapter 27) can be automatically saved. A file of "exhaustive pairs", made up of each state and its successor, can be created (see section 29.5).

Various data including mean entropy and entropy variance of space-time patterns can be automaticaly generated and saved. This allows a sorted sample of CA rules to be created, discriminating between order, complexity and chaos (see section 33. A large collection of complex rules, those featuring "gliders" or other large scale emergent structures, can be assembled. Pre-assembled files of 1d CA rules sorted by this method[16] are provided with DDLab, for k=5, 6 and 7 (see section 32.5.1).

2.9.3 Filing the screen image

The screen image is saved and loaded using an efficient home-made compressed format which is only applicable within DDLab (see section 5.4.

In DOS, to save the image in a standard format (and print to any printer), use a "stay resident screen grabber". A number of specialist screen grabbers are available, and others are part of "paint" programs. Alternatively run DDLab as a DOS application in Microsoft Windows and use their "paint" screen grabber.

To save the image in Unix or Linux use a program such as XView to grab the DDLab window or part of it, and save it in many standard formats. This is how the figures in this manual where produced as PostScript files.

2.9.4 Printing the screen image
(see section 5.5)

The screen image can be printed at any time directly from DDLab. In DOS, the following printers have been tested and seem reliable: Epson MX-82 dot-matrix printer, Cannon BJC 4000 bubble jet. Printing to other printers is worth a try but is unpredictable.

In the Unix or Linux the screen can be printed from options within DDLab as a PostScript file to a laser printer. Alternativly use XView to grab the image and print.

2.10 Mutations

A wide variaty of network "mutations", as well as changes in presentation, can be be made, many on-the-fly while the simulation is running.

2.10.1 Running Forward

When running forward, key-press options allow changes to be made to the network and presentation on-the-fly (see chapterDrawing space-time patterns, and changes on-the-fly). This includes "mutations" to wiring, rules, current state, and size. A number of 1d "complex" rules (with glider interactions) can be set for $k = 5$, 6 and 7 (see section 32.5.1).

2.10.2 Running Backward

When running backward, and attractor basins are complete, a key press will regenerate the attractor basin of a mutant network. Various mutation options can be pre-set (see chapter 28) including random bit-flips in rules and random rewiring of a given number of wires. Sets of states can be specified and highlighted in the attractor basin to see how mutations affect their distribution (see chapter 34).

2.11 Quantitative, statistical and analytical measures

Some of the measures and data on network dynamics available in DDLab are listed below. In most cases this information can be displayed graphicaly.

2.11.1 Behaviour parameters

The following static parameters measured on rule look-up tables are available (see section 16.9.5).

- The λ parameter, and equivalent P parameter (see section 16.9.2).
- The Z parameter (see section 16.9.3).
- The (weighted) average λ and Z for a mixed rule network (see section 17.8.1).
- The frequency of canalizing "genes" and inputs (see chapter 15 for a rule-mix network, section 18.7 for single rules).

λ, Z, and canalization can be set to any arbitrary level,

2.11.2 Network connectivity

The following measures on network connectivity, i.e. the wiring, are available,

- Average k (inputs), number of reciprocal links, and self links (see section 17).
- Histograms of the frequency distribution of inputs (i.e. k), outputs, or both (i.e all connections) in the network (see section 17.8.13).
- The recursive inputs/outputs to/from a network element, whether direct or indirect, showing the "degrees of separation" between elements (see sections 17.4.7 and 17.4.8).
- The network graph (see section 20), where the wiring is analyzed in two ways, as an adjacency matrix or table, and derived from this, as a a graph that can be analyzed in various ways, and rearranged and unraveled, including dragging vertices and defined components to new positions with "elastic band" edges.

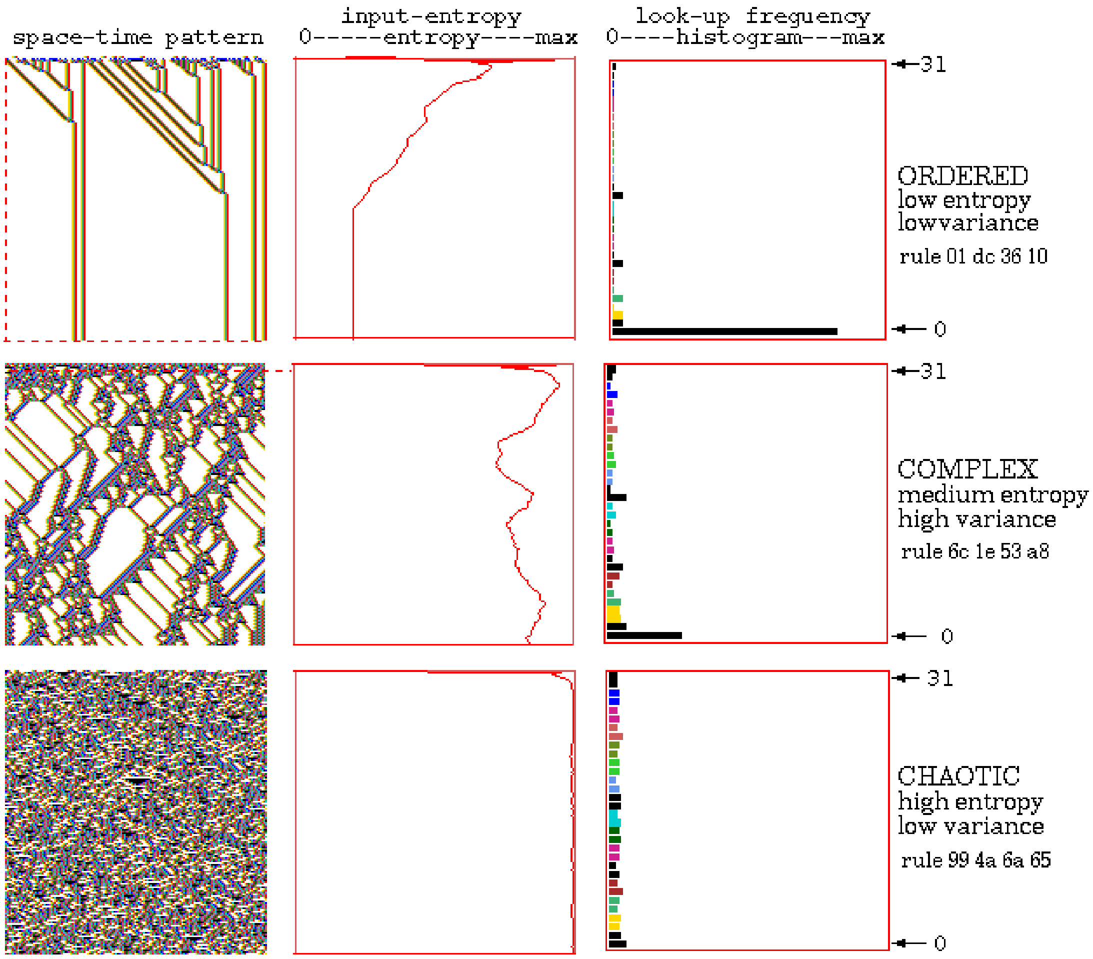
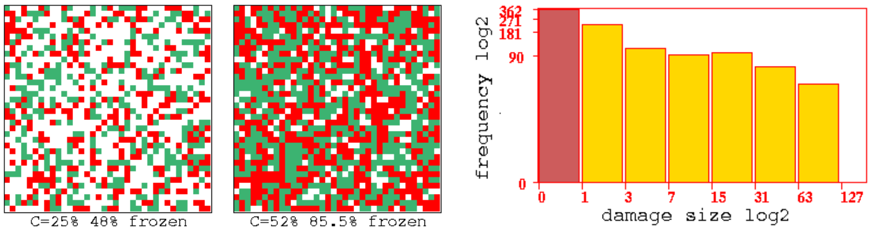


Figure 2.4: Typical 1d CA space-time patterns showing ordered, complex and chaotic dynamics (n=150, k=5,rule numbers shown in hex). Alongside each space-time pattern is a plot of the input-entropy, where only complex dynamics (*centre*) exhibits high variance caused by glider collisions.



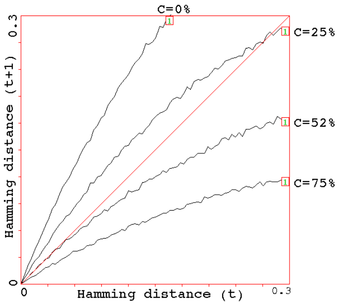


Figure 2.5: Order-chaos measures for a RBN 36×36, $k = 5$. C = the percentage of canalizing inputs in the randomly biased network. *top left*: frozen elements that have stabilized for 20 time-steps are shown, 0s-green, 1s red, otherwise white, for C=25% and 52%. *top right*: the log-log "damage spread" histogram for C=52%, sample size about 1000. *left*: the Derrida plot for C=0%, 25%, 52%, and 75%, for 1 time-step, H_t=0-0.3, interval = 5, sample for each H_t = 25.

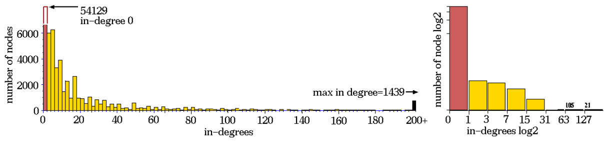
2.11.3 Measures on local dynamics

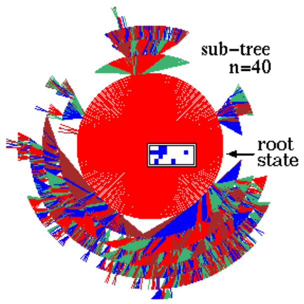
The following measures on local dynamics, i.e. running the system forward from some initial state, its space-time patterns or trajectories, are available,

- A rule-table lookup frequency histogram, which can be toggled between 2d and 3d to include a time dimension (see section 31.9).
- The entropy of the lookup frequency over time (see section 32.11.2).
- The variance of the entropy, and an entropy/density graph, where complex rules have their own distinctive signatures (see section 32.11.4).
- A plot of mean entropy against the variance of the entropy for an arbitrarily large sample of CA rules, which allows ordered, complex and chaotic rules to be classified automatically (see chapter 33), also shown as a 2d frequency histogram (see section 33.6.2). Ordered, complex and chaotic dynamics are located in different regions allowing a statistical measure of their frequency. In addition the rules can be sorted by entropy variance allowing complex rules to be found automatically.
- The pattern density over time (see section 31.9).
- The activity/stablity of network elements (see section 32.10). Frozen islands, the fraction of "genes" that have not changed over the last x generations, the fraction of frozen 0s and 1s, and "genes" colored according the fraction of time they have been "on" (i.e. 1) in the same window of x time-steps, falling into preset "frequency bins".
- The damage spread, or pattern difference, between two networks in 1d or 2d. A histogram of damage spread can be automaticaly generated for identical networks with initial states differing by 1 bit (see section 31.15).

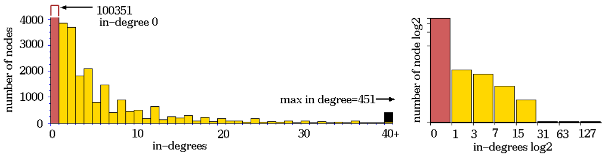
- The “Derrida plot”, and Derrida coefficient, analogous to the Liapunov exponent in continuous dynamical systems, measures how pairs of network trajectories diverge/converge in terms of their Hamming distance. This indicates if a random Boolean network is in the ordered or chaotic regime (see chapter 22).

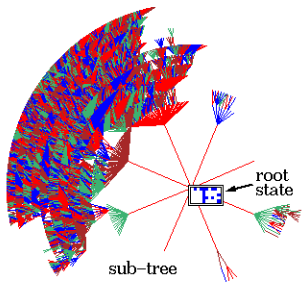
- A scatter plot of successive iterations in a 2d phase plane, the “return map”, showing a fractal structure, especially for chaotic rules (see section 32.11.6).



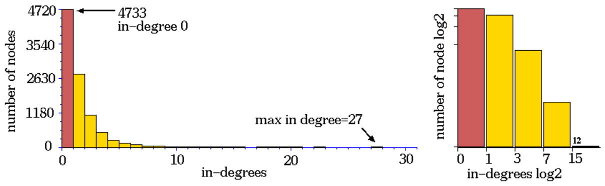


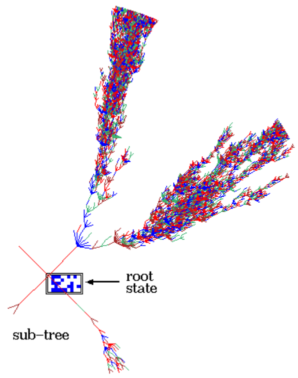
Ordered dynamics. Rule 01dc3610, n=40, Z=0.5625, λ_{ratio}=0.668. *right:* The complete sub-tree 7 levels deep, with 58153 nodes, G-density=0.931.





Complex dynamics. Rule 6c1e53a8, n=50, Z=0.727, λ_{ratio}=0.938. *right:* The sub-tree, stopped after 12 levels, with 144876 nodes, G-density=0.692.





Chaotic dynamics. Rule 994a6a65, n=50, Z=0.938, λ_{ratio}=0.938. *right:* The sub-tree, stopped after about 75 levels, with 9446 nodes, G-density=0.487.

Figure 2.6: Subtrees of ordered-complex-chaotic CA. The space-time patterns of the rules are shown in figure 2.4. The in-degree histogram of a typical sub-tree shown in normal and log-log form.

2.11.4 Measures on global dynamics

The following measures on global dynamics, i.e. attractor basins, are available,

- Data on attractor basins. The number of basins in the basin of attraction field, their size, attractor period and branching structure of transient trees. Details of states belonging to different basins, subtrees, their distance from attractors or the subtree root, and their in-degree (see chapter 27).

- A histogram of attractors (or skeletons) showing the frequency of arriving at different attractors from random initial states. This provides statistical data on the basin of attraction field for large networks. The number of basins, their relative size, period, and the average run-in length can be measures statistically (see section 31.17). An analogous method shows the frequency of arriving at different "skeletons", partly frozen patterns.

- Garden-of-Eden density plotted against the λ and Z parameters (see section 24.5), and aginst network size (see section 24.5.4).

- A histogram of the in-degree frequency of attractor basins or subtrees (see section 24.3).

- The state-space matrix, a scatter-plot of the left half against the right half of each state bit string, using colour to identify different basins, or attractor cycle states (see section 24.2).

- The attractor meta-graph, an analysis of the basin of attraction field tracking where all possible 1-bit flips to attractor states end up, whether to the same, or to which other, basin (see section 20). The information is presented in two ways, as a jump-table: a matrix showing the jump probabilities between basins, and as a meta-graph: a graph with weighed vertices and edges giving a graphic representation of the jump-table. The meta-graph itself can be analyzed and manipulated in various ways, and rearranged and unraveled, including dragging vertices and defined components to new positions with "elastic band" edges.

2.12 Reverse algorithms

There are three different reverse algorithms for generating the pre-images of a network state.

- An algorithm for 1d CA, or networks with 1d CA wiring but heterogenious rules.

- A general algorithm for random Boolean networks, which also works for the above.

- An exaustive algorithm that works for any "random mapping" inluding the two cases above.

The first two reverse algorithms (see section 29.9) generate the pre-images of a state directly; the speed of computation decreases with both neighborhood size k, and network size. The speed of the third exhaustive algorithm (see section 29.5) is largely independent of k, but is especially sensitive to network size.

The method used to generate pre-images will be chosen automatically, but can be overridden. For example, a regular 1d CA can be made to use either of the two other algorithms for benchmark purposes and for a reality check that all methods agree. The time taken to generate attractor basins is displayed in DDLab. For the basin of attraction field a progress bar indicates the proportion of states in state-space used up so far.

2.12.1 1d CA wiring

The CA reverse algorithm applies specifically for networks with 1d CA wiring (local wiring) and homogeneous k, such as 1d CA, though the rules may be heterogeneous. This is the most efficient thus fastest algorithm, described in [12, 19]. Furthermore, compression of 1d CA attractor basins by rotation symmetry (see section 26.1) speeds up the process.

2.12.2 Non-local wiring

Any other network architecture, with non-local wiring, will be handled by a slower *general* reverse algorithm described in [13, 19]. A histogram revealing the inner workings of this algorithm can be displayed. Regular 2d or 3d CA will also use this general reverse algorithm though in principle more efficient algorithms that take advantage of 2d or 3d local wiring could be devised. However, compression algorithms will come into play in 2d to take advantage of the many rotation symmetries on the torus[1].

2.12.3 Exhaustive testing

A third, brute force, reverse algorithm first sets up a mapping, a list of "exhaustive pairs" of each state in state-space and its successor (this can be saved). The pre-images of states are generated by reference to this list. The exhaustive testing method is restricted to small systems because the size of the mapping increases exponentially as 2^n, and scanning the list for pre-images is slow compared to the direct reverse algorithms for CA and RBN. However, the method is not sensitive to increasing neighborhood size k, and is useful for small networks with large k. Exhaustive testing is also used for sequential updating ((see section 29.7). The basin of attraction field of the 4x4 "game-of-Life", where k=9, generated by this method took about 30 minutes on my 66MHz 486).

2.13 Random map

The random mapping routine (see section 29.6) creates a list of "exhaustive pairs", assigning a successor state at random to each state in state space, possibly with some bias (rules and wiring previously set are ignored). The attractor basins are reconstructed by reference to this random map with the exhaustive testing algorithm. The space of random maps for a given system size corresponds to the space of all possible basin of attraction fields and is the super-set of all other deterministic discrete dynamical systems.

2.14 Sequential updating

(see section 29.7)

By default, network updating is synchronous, in parallel. DDLab also allows sequential updating, both for space-time patterns (see sections 32.3) and attractor basins (see section 29.7.5). Default orders are forwards, backwards or a ramdom order, but any specific order can be set out of the $n!$ possible orders for a network of size n. The order can be saved/loaded from a file.

---

[1]Compression does not apply for a regular 2d CA where k=6 or k=7, which has a triangular grid, or for 3d CA, as the algorithms to take account of these symmetries have not been resolved.

2.14.1 Neutral order components

(see section 29.8)

An algorithm in DDLab computes the neutral order components (limited to network size $n \leq 12$). These are sets of sequential orders with identical dynamics. DDlab treats these components as subtrees generated from a root order, and can generate a single component subtree, or the entire set of components subtrees making up sequence space (the neutral field) which are drawn in an analogous way to attractor basins.

2.15 Sculpting attractor basins

(see section 34)

Learning and forgetting algorithms allow attaching and detaching sets of states as predecessors of a given state by automatically mutating rules or wiring couplings. This allows "sculpting" the attractor basin to approach a desired scheme of hierarchical categorization. Because any such change, especially in a small network, usually has significant side effects, the methods are not good at designing categories from scratch, but might be useful for fine tuning a network which is already close to where its supposed to be.

When an attractor basin is complete, within the learning routine, a "target" state, together with a number of "aspiring pre-images" (predecessors) can, be selected. These states may be just highlighted in successive mutant attractor basins, or the learning/forgetting algorithms will attempt to attach/detach the aspiring pre-images to/from the target state, and can be set for either rule-table bit-flips or wire moves. In fact the bit-flip method cannot fail. New attractors can be created and sub-trees transplanted. The result of learning/forgetting, including side effects, will be apparent in the new attractor basins. The algorithms, and their implications are described in[13].

More generally, a very preliminary method for reverse engineering a network, also known as the inverse problem, is included in DDLab, by reducing the connections in a fully connected network to satisfy an exhaustive map (for network sizes $n \leq 13$, see section 18.12). The inverse problem is finding a minimal network that will satisfy a full or partial mapping (i.e. fragments of attractor basins such as trajectories).

2.16 Hardware and software requirements

Compiled versions of DDLab currently run on the following platforms:

- **DOS/PC**: compiled with Watcom C version 11.
- **Linux/PC**: RedHat version 7.1, compiled with gcc.
- **Unix/XWindows-Sun**: Sun Solaris, compiled with gcc.
- **Irix/SGI:** for IRIX 6.5.11, MipsC 7.3.1.2[2].

[2]Thanks to Oskar Itzinger (oitzinger@opec.org) for the Irix port.

Compiled versions of DDLab for other platforms may also be available, for example hpux/X11/HP[3] and Debian Linux[4]. See one of the following DDLab web sites for current information,

```
www.ddlab.com
www.santafe.edu/~wuensch/ddlab.html
www.cogs.susx.ac.uk/users/andywu/ddlab.html
```

I am reliably informed that the DOS version of DDLab runs on a Mac using Virtual PC.

2.16.1 Source Code

The DDLab source code is written in C. It may be made available on request, subject to various conditions, for porting to other platforms or adding new features.

2.16.2 DOS Platform

The DOS version runs on a 386 PC or higher, with maths co-processor, mouse (optional), and with VGA or SVGA graphics, giving a choice of screen resolution of 640×480, 800×600 or 1024×768. The code is compiled with the Watcom C32 compiler version 11 (and the Rational Systems DOS extender) giving access to extended memory. The supplied Watcom file `dos4gw.exe` must be in the same directory as `ddlab.exe`. Note that DDLab may also be set up and run as a DOS application under Microsoft Windows95[5], but is probably more stable when run in pure DOS mode.

2.16.3 Unix/XWindows platform

Compiled in gcc for SUN Solaris 2.5, with "static" set so that missing library problems should not occur.

However, the libraries `libX11.so.6.1` and `libsunmath.so.1` do need to be accessible in your system, they usually are. If they are missing they can be downloaded from the DDLab web site and should be installed in the same directory as DDLab, or some other appropriate directory. If your system *still* cant find them, you should amend the `LD_LIBRARY_PATH` to point to the library files (your systems admin might help at this point), or give the following command,

```
setenv LD_LIBRARY_PATH ~your_username/directory_name.
```

2.16.4 Linux platform

Compiled with gcc in RedHat version 7.1. Runs in the same way as the Unix/XWindows version.

2.16.5 Irix/SGI platform

Compiled with with -n32 and -O2 under IRIX 6.5, and requires the X Window System. Runs in the same way as the Unix/XWindows version.

---

[3]Rick Riolo (rlriolo@umich.edu) made a hpux/X11/HP port in 1996
[4]Valerio Aimale (valerio@biosgroup.com) made a Debian Linux port in 1999
[5]this also applies to Windows 4.1 and Windows98

Chapter 3

Accessing DDLab

This chapter gives detailed instructions for downloading, unzipping, unpacking, and running DDLab, for DOS, Unix, Linux, and Irix. Information is also given about the DDLab web site, mailing list, license, copyright, disclaimer, and Discrete Dynamics Inc., the company created to market and support DDLab.

3.1 The DDLab web site

For the latest version check the DDLab web site located at one of the following,

```
www.ddlab.com
www.santafe.edu/~wuensch/ddlab.html
www.cogs.susx.ac.uk/users/andywu/ddlab.html
```

3.2 Versions of DDLab

The current and older versions of DDLab and documentation are listed below. The older versions may be available only at the Santa Fe Institute.

- July 2001 DDLab version 23, sub-directory `dd_2001_july`.

  `ddlab_dos_23.tar.gz`: DOS/PC.
  `ddlab_unix_23.tar.gz`: Unix/XWindows/Sun, compiled for Solaris 2.5.
  `ddlab_linux_23.tar.gz`: Linux/PC, compiled for Red Hat 7.1.
  `ddlab_irix_23.tar.gz`: Irix 6.5 for SGI.

- Feb 1999 DDLab version 22, sub-directory `dd_1999_feb`.

  `ddlab_22.zip`: DOS/PC.
  `ddlab_unx_22.tar.gz`: Unix/XWindows/Sun, compiled for Solaris 2.5.
  `ddlab_lnx_22.tar.gz`: Linux/PC, compiled for Red Hat 5.1.

- Sept 1997 DDLab version 21, sub-directory `dd_1997_sept`.

  `ddlab_21.zip`: DOS/PC.
  `ddlabx_21.tar.gz`: Unix/XWindows/Sun, compiled for Solaris 2.5.
  `ddlabx_linux_21.tar.gz`: Linux/PC, compiled for Linux 1.99 ELF.

- March 1996 version version of DDLab, sub-directory `dd_1996_march`.

  `ddlab_20.zip`: DOS/PC.
  `ddlabx_20.tar.gz`: Unix/XWindows/Sun, compiled for OS 4.1.

- July 1995 version of DDLab, sub-directory `dd_1995_jul`.

  `ddlab_10.zip`: DOS/PC.

- The DDLab manual, sub-directory `dd_manual`.

  `ddop2001.ps.gz`: valid for DDLab version 23, July 2001, PostScript format.
  `ddop2001.pdf`: valid for DDLab version 23, July 2001, pdf format.
  `ddman_20.zip`: valid for DDLab version 20, March 1996, Word for Windows format.

3.3 FTP from the command line

DDLab files can be found at the following FTP sites,

`ftp://ftp.santafe.edu/pub/wuensch/` ...at SFI
`ftp://ftp.cogs.susx.ac.uk/pub/users/andywu/` ...at COGS

To download versions of DDLab and documentation by anonymous FTP from the command line, first change to the directory where DDLab is to be installed, then follow a typical FTP session as set out below,

`ftp ftp.santafe.edu`	at SFI, or `ftp ftp.cogs.susx.ac.uk` at COGS.
`name: anonymous`	
`password:`	your complete email address, the prompt changes to `ftp>`
`cd /pub/wuensch`	to change directory at SFI,
	or `cd /pub/users/andywu/` at COGS
`ls`	to list the sub-directories
`cd dd_2001_july`	to change to a sub-directory, for example
`ls`	to list the files
`bin`	set for binary mode
`get ddlab_linux_23.tar.gz`	to get the required file, for example
`cd ..`	(cd, space, dot dot), for the higher level directory
`quit`	to quit FTP

3.4 DDLab documentation

The manual you are reading relates to the latest DDLab version 23, July 2001. The manual will be updated from time to time, check the DDlab web site for update news. The manual is available in the sub-directory `dd_manual`, in the following formats,

`ddop2001.ps.gz` ... this will unzip to give `ddop2001.ps`,
a PostScript file to be read with GhostView.

`ddop2001.pdf` ... a `pdf` file to be read with Acroread.

The old manual is still available in the file `ddman_20.zip` which will unzip (with PKUNZIP) to `ddman.doc`, a Word for Windows file. This un-illustrated reference manual is only valid for the DDLab version 20, March 1996, and prior versions.

3.5 Unzipping and running - DOS

The latest DOS wersion of DDLab, `ddlab_dos_23.tar.gz`, July 2001, can be unzipped with Winzip from Windows, or with `gunzip` and `tar -xvf` from Unix/Linux as in section 3.6.

Previous versions (`.zip`) were zipped with PKZIP and must be unzipped with PKUNZIP. A net search will list sites that describe and provide PKZIP.

The DOS version will unzip to give a number of files, including,

`ddlabx23.exe` ... the program *(for example).*
`dos4gw.exe` ... the DOS extender, access to extended memory.

Keep all the files together in their own directory.

The default background is black for version 23, white for previous versions. For best results DDLab should be run in pure DOS, or in a full screen DOS window in Windows. In a small DOS window only 640x480 resolution is allowed. To toggle between a small DOS window and full screen, enter `Alt+Enter`. To run the program, enter `ddlabx23` at the DOS prompt for a black background and 640x480 pixel resolution.

You can also add the following program parameters for a different graphics setup (this can also be changed later).

`-w` ... for a white background (in previous versions `-b` for a black background.)
`-m` ... for 800x600 resolution (Feb 1999 and later versions only).
`-h` ... for 1024x768 resolution.

For example, for a white background and 1024x768 enter `ddlabx23 -w -h`.

3.6 Unzipping and running - Unix, Linux, Irix

Place the `.tar.gz` file in its own directory. To unzip the `.gz` file, then unpack the `.tar` file, follow the example below,

`gunzip ddlab_linux_23.tar.gz`	... to unzip the `.gz` file
`tar -xvf ddlab_linux_23.tar`	... to unpack the `.tar` file

This will give a number of files, including the executable, `ddlabx23` (for example). Keep these files in the same directory. To run the program enter `ddlabx23 &` (or `./ddlabx23 &` on some Unix systems) , i.e. the name of the executable followed by `&` (to retain control of the xterm window).

The default background is black for version 23, white for previous versions. This can be changed with a program parameter, `-w` for version 23, `-b` for previous versions.

3.6.1 Unix library files

DDLab for Unix is compiled with "static" set, so that missing library problems should not occur. However, the libraries `libx11` and `libsunmath` do need to be in your system, they usually are. If they are missing, they can be downloaded from the file,

```
unix_libs.tar.gz
```

which will unzip and unpack as described above to give the following files,

```
libX11.so.6.1
libsunmath.so.1
```

These files should be installed in the same directory as DDLab.

3.7 Extra data files

These files, common to all platforms, contain data used by DDLab, including samples of "complex" 1d CA rules which feature interacting gliders, and should be in the same directory as the DDLab executable file.

`glider5.r_s`	... about 62 complex k=5 rules.
`glider6.r_s`	... about 31 complex k=6 rules.
`glider7.r_s`	... about 31 complex k=7 rules.
`five5ss.sta`	... a sorted sample of 17680 k=5 rules.
`six5ss.sta`	... a sorted sample of 15425 k=6 rules.
`sev5ss.sta`	... a sorted sample of 14221 k=7 rules.
`pento.eed`	... the "rpentomeno" 2d pattern, to seed interesting "game-of-life" dynamics.

3.8 The Quick Start Examples

Chapter 4 gives brief "quick start" examples for a number of common functions. Its a good idea to try these first get the flavour of DDLab before reading the detailed manual.

3.9 The DDLab mailing list

To join the DDLab mailing list, email `majordomo@santafe.edu` with "subscribe ddlab" in the body of the message (conversely "unsubscribe ddlab"). Subscribers may post questions, answers, comments and discussion of DDLab issues to `ddlab@santafe.edu`. The DDLab mailing list is archived at `www.santafe.edu/~wuensch/archive/`.

3.10 License and Copyright

3.10.1 License

DDLab remains free shareware for personal, non-commercial, users only. Any other users, including commercial users, companies, government agencies, research or educational institutions, must register and pay the license fee.

To register, email your details to `ddlab@ddlab.com`. Details of the current license fees may be found at `www.ddlab.com/ddinc.html`.

At present only compiled versions of DDLab can be downloaded from the DDLab home page `www.ddlab.com`. However, the DDLab source code may be made available on request, subject to various conditions, for porting to other platforms or adding new features.

3.10.2 Copyright

DDLab is copyright (c) 1993-2001, Andrew Wuensche (`andy@ddlab.com`).

3.10.3 Disclaimer

No warranty is made for DDLab or its performance, and no responsibility or liability whatsoever is accepted to anyone for the consequences of using it.

3.11 Discrete Dynamics Inc.

Discrete Dynamics, Inc. was created to market network simulation software, in particular DDLab, and to allow the continued development and support of DDLab in the future. The company will also undertake consultancy, research, development and applications in the area of discrete dynamical networks.

Chapter 4

Quick Start Examples

This chapter briefly describes the DDLab graphical use interface, and gives a number of examples of DDLab functions. Try these examples first, to get the flavor of DDLab, before reading the detailed program reference (chapter 5 onwards).

4.1 The DDLab screen

In Unix, Linux, Irix, or Windows(3.1, 95, 98), DDLab occupies a window within the monitor screen, in DOS the whole screen is occupeid. For simplicity we will refer to the DDLab "screen", and various panels which appear from time to time within the screen as "windows". The location of a window within the screen is usually indicated, for example "bottom", "top right" etc.

When DDLab is run with no program parameters, the screen appears with a black background. The program parameter `-w` gives a white background, i.e. enter `ddlabx23 -w`. Descriptions of colors in this manual assume a white background.

A title bar is displayed across the bottom of the screen. A series of prompts are presented to set up the network, functions to be performed, presentation. These prompts appear either in a main sequence for the most common settings, or in various windows that automatically open up.

The mouse cursor (if detected) is used to set bits in rule-tables and network states, for "drawing" bit patterns in 2d networks, for dragging nodes in the network graph and meta-graph, and for some other functions, but most user inputs are from the keyboard.

4.1.1 User Input

The flashing cursor (usually green) prompts for input. Enter appropriate input from the keyboard.

To revise the input, press **q**, **backspace**, or right mouse button.

To accept the input, and move on to the next prompt or routine, press **return** or left mouse button. If no input was entered, or if the input was inappropriate, a default input is automatically selected.

4.1.2 Backtrack

To backtrack to the preceding prompt, to revise, or interrupt a running routine such as space-time patterns or attractor basin, press **q**, or right mouse button.

You can backtrack to any stage in the prompt sequence with **q** (or right mouse button), eventually to exit the program.

4.1.3 Quitting DDLab

To quit DDLab immediately (except in DOS) enter **Ctlr-q** at any prompt, followed by **q**. Otherwise backtrack with **q** to the start of the program, then enter **q** to exit.

4.1.4 Skipping Forward

At some points in the prompt sequence, its possible to skip forward, to avoid a succession of prompts for special settings. When the following top center banner is visible,

accept defaults-d ...enter **d** to skip forward.

4.1.5 The graphics setup - DOS

Its recomended that you run DDLab in pure DOS mode. In Windows95 use the option "Restart the computer in MS-DOS mode". In a DOS window in Windows95, run DDLab in the default low resolution only. If you expand to full screen (toggle **Alt-Enter**) you may use higher resolutions but the mouse cursor will probably not be visible. Be sure to revert to low resolution before changing from full screen back to the DOS window.

In DOS, the background can be changed between black and white, and the resolution between 640×480, 800×600, and 1024×768 pixels, given the necessary monitor, graphics card and driver.

The DDLab screen will start up at a resolution of 640×480 (VGA) with a black background, if no program parameters are set. Program parameters may be added for SVGA as follows: `-h` (for high, 1024×768) and `-m` (for medium, 800×600), i.e. for high resolution start DDLab by entering `ddlabx23 -h` at the DOS prompt. For a white background enter `ddlabx23 -w`, or `ddlabx23 -h -w` for both high resolution and a white background.

To change the resolution or background after DDLab has started, at the first prompt select **g** for graphics. A graphics setup screen will appear. Enter **b** to toggle the background. Other options allow changing the resolution, font size and text line spacing and cursor flash speed.

4.1.6 The graphics setup - Unix, Linux, Irix

The screen will start with a black background if no program parameters were set, and with a resolution automatically selected to comfortably fit the monitor. For example, on my 1024×768 laptop running linux, the resolution of the DDLab window is 925×694.

To change the background after DDLab has started, at the first prompt select **g** for graphics. A graphics setup screen will appear. Enter **b** to toggle the background. Other options allow changing the resolution, font size and text line spacing and cursor flash speed. The screen can also be resized with the mouse in the usual way.

4.2 Basin of attraction fields

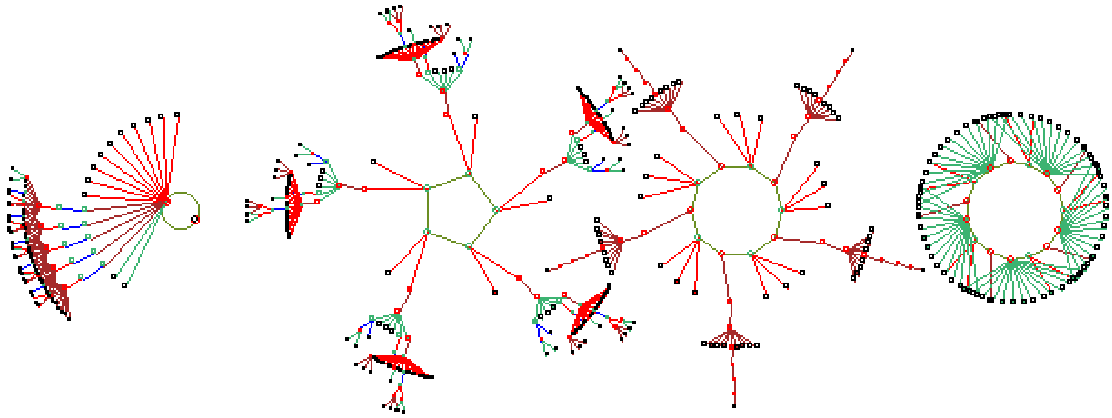


Figure 4.1: A basin of attraction field of a Cellular Automaton, $k = 3$ (rule 9), $n = 10$, equivalent basins suppressed.

From the first prompt keep accepting defaults with **return** or left mouse button (about 13 presses), until the top center **output parameters** banner appears, and a top right window with a list of options starting with **accept all basin defaults -d**. Enter **d** to skip these special options.

1. A new top right prompt window appears. Enter **return**.
2. The basin of attraction field will be generated for a 1d 3-neighbor CA rule, n=10. The rule (chosen at random by default) appears in a window at bottom of the screen. A top right window shows brief data on the field once it is generated. A progress bar below this window shows the proportion of state-space as it is used up. Vertical lines on this bar indicate the states used to seed the basins.
3. A prompt window appears top left. Enter **return**.
4. Another basin of attraction field is generated, a one bit "mutant" of the previous rule, with corresponding data. This process can continue indefinitely (it can be set on automatic).
5. Enter **q** to interrupt and backtrack up the prompt sequence.
6. Try this routine again with an increased network size n and neighborhood size k.
   At the prompt **Network size (length) max 31, default 10:** enter 14 (for example).
7. At the second prompt after that
   Neighborhood size k: mixed-m, or enter 1-13 (def 3): enter 5.
8. Then repeat the steps as above. The basins of attraction will take longer to generate. Their scale and position can be fine-tuned with the special options that were skipped.

4.3 "Backwards" space-time patterns, and state-space matrix

While the attractor basins are generating, various display settings can be changed on-the-fly. These are indicated on the right of the bottom title bar. If the attractor basin generates to fast, backtrack to slightly increase the network size n or neighborhood k.

1. Enter **s** to toggle the "backwards" space-time pattern on-off, and see predecessors (pre-images) being generated on the left of the screen as a space-time pattern. Initially the attractor states will be displayed, then each state (black) and its set of pre-images (red). Expand or contract the scale of the space-time pattern with **e** and **c**. Toggle scrolling with #.
2. Enter **m** to toggle the display of the state-space matrix in the lower right corner. This reveals interesting symmetries. Different colors represent states in different basins.
3. Enter **q** to interrupt and backtrack up the prompt sequence.

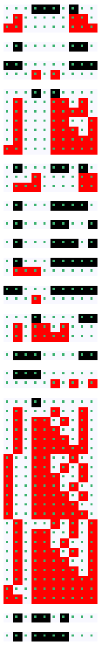


Figure 4.2: Backwards space-time patterns relating to the basin of attraction field of the Cellular Automaton in figure 4.1. Space across, time top down. The red and white bit patterns are the predecessors of black and white bit patterns.

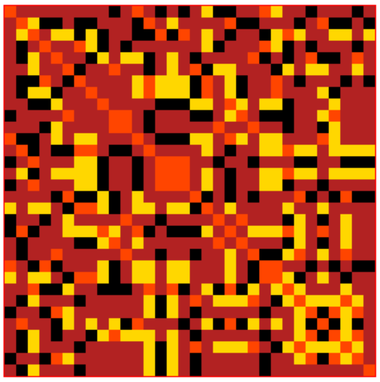


Figure 4.3: The state-apace matrix represents state-space, plotting the left half of each state bitstring against the right half. Colors represent different basins of attraction in figure 4.1.

4.4 Basin of attraction fields for a range of network sizes

Backtrack with **q** (or right mouse button) to the start of the program.

1. At the second prompt, **range of network size-r, else-ret:** enter **r**.
2. Enter **return** until the top center **output parameters** banner appears, then **a** to restore all defaults, then **d** to skip further special options.

3. Enter **return** to start a *range* of CA basin of attraction fields (with the same rule), for increasing sizes from 5 to 12.

4. Enter **return** for the next "mutant" CA rule.

5. Toggle the display of "backwards" space-time patterns and the state-space matrix as described in section 4.3

6. Enter **q** to interrupt and backtrack up the prompt sequence.

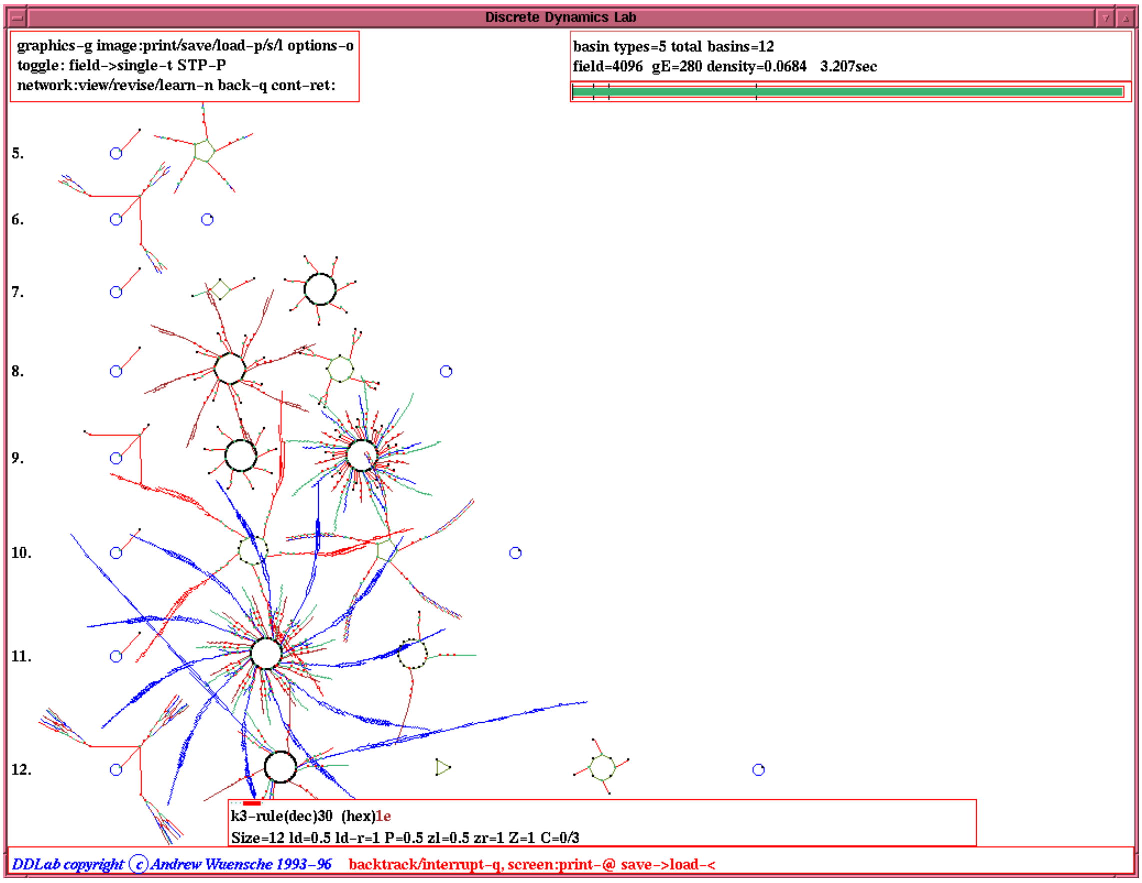


Figure 4.4: Basin of attraction fields for a range of network size 5-12. $k = 3$, rule=30

4.5 A single basin of attraction

Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt
   forward only/single basin/subtree-s (default field): enter **s**.

2. At the **Neighborhood size...** prompt, enter 4.

3. Enter **return** until the top center **output parameters** banner appears, then **a** to restore all defaults, then **d** to skip further special options.
4. Enter **return** in response to further prompts in top right windows.
5. A singe basin for a CA, size 14, is generated.
6. Enter **return** for the next mutant.
7. Toggle the display of the "backwards" space-time patterns with **s** and the state-space matrix with **m** as described in 4.3.
8. Enter **q** to interrupt and backtrack up the prompt sequence.

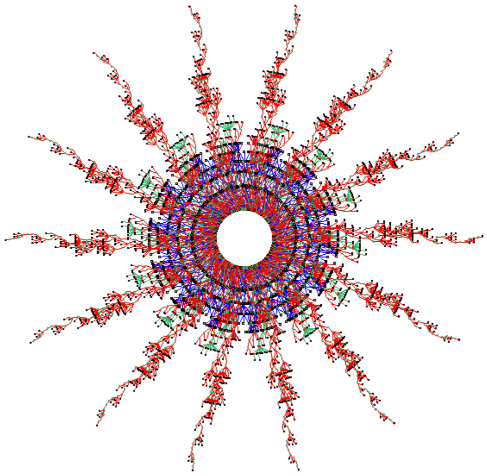


Figure 4.5: A single basin of attraction with 7154 states, and an attractor period of 714, $k = 4$, hex rule=66 e2, $n = 14$.

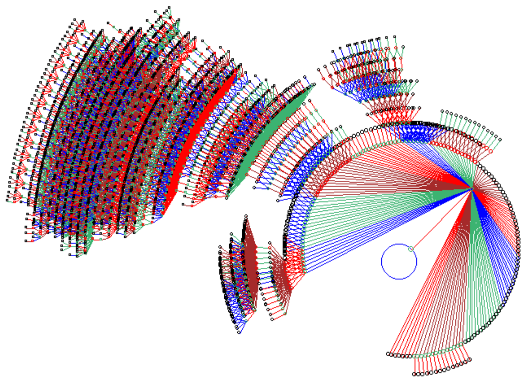


Figure 4.6: A single basin of attraction with 15836 states and an attractor period of one, a point attractor, $k = 4$, hex rule=7b e6, $n = 14$.

4.6 A subtree

Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt **forward only/single basin/subtree-s (default field):** enter **s**.
2. Enter **return** in response to further prompts until the **Network size...** prompt, select 25.
3. At the **Neighborhood size...** prompt, select 5.
4. Enter **return** in response to further prompts until the **Select SEED...** window appears. Enter **r** for a "random seed" then **a** to set *all* cells at random.
5. Enter **return** until the top center **output parameters** banner appears, then **a** to restore all defaults, then **d** to skip further special options.

6. At the prompt **backward for subtree-b, forward for basin (def):** select **b**.

7. At the next prompt, **forwards before backwards?**
   how many steps (default 0, max 4096): select 2.

   This runs the CA forward by 2 time-steps (from the "seed"), before running backward from the state reached. The original randomly selected seed is likely to be a "garden-of Eden" state with no predecessors, so not much use as the root of a sub-tree.

8. Enter **return** in response to further prompts in top right windows. The subtree is generated with its "root" state highlighted as a bit pattern. To generate a bigger sub-tree, enter a greater number of forward time-steps at the previous prompt. However, this might reach an attractor state. In this case the whole basin will be generated with the message **subtree=basin** in the top right data window. If this is taking too long, enter **q** to interrupt and backtrack to reduce the number of forward time-steps.

9. Enter **return** for the next mutant.

10. While the subtree is being generated, toggle the display of "backwards" space-time patterns with **s** and the state-space matrix **m** as described in 4.3.

11. Enter **q** to interrupt and backtrack up the prompt sequence.

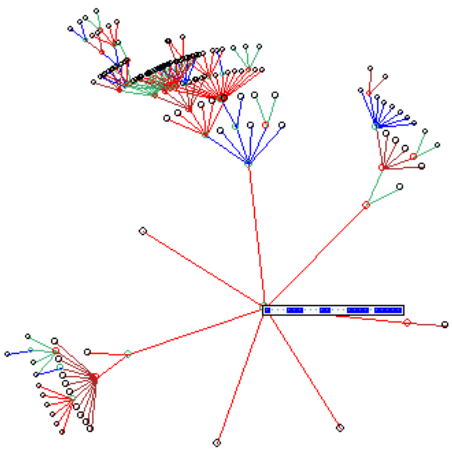


Figure 4.7: A subtree generated from the bit pattern at the center. 1d CA, $k = 5$, hex rule f4 0e 00 ef, $n = 25$.

4.7 1d Space-time patterns

Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt **forward only/single basin/subtree-s (default field):** enter **s**.

2. At the **enter cell scale in pixels...** prompt, select 1, the smallest scale.

3. At the **Network size...** prompt, select 150.

4. At the **Neighborhood size...** prompt, select 5.

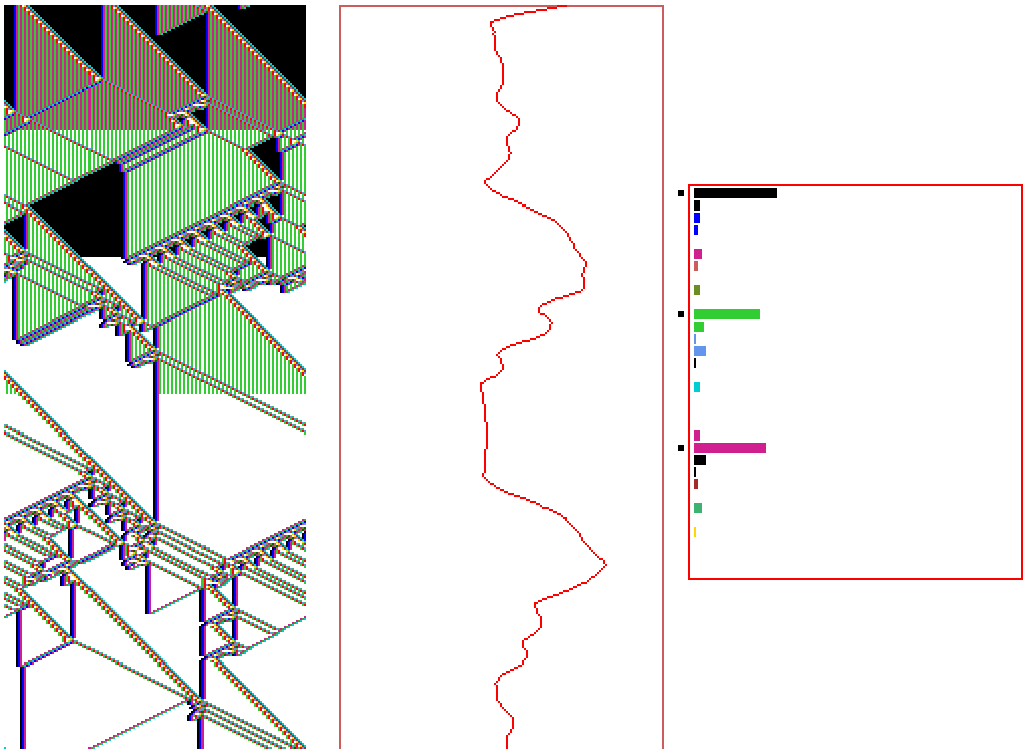


Figure 4.8: A space-time pattern of a "complex" 1d CA, $k = 5$, hex rule e9 f6 a8 15, $n = 150$. About 360 time-steps, including some analysis shown by default. *left*: The space-time pattern colored according to neighborhood, and progressively "filtered" at three times with key **f**, suppressing the background domain to show up "gliders" more clearly. *center*: The input-entropy/time plot. *right*: The lookup frequency histogram for the last time step shown.

5. Enter **return** until the top center $\boxed{\textbf{output parameters}}$ banner appears, then enter **s** for **space-time pattern only**.

6. At the next prompt, **FORWARDS ONLY options:...**, the top center $\boxed{\textbf{accept defaults-d}}$ banner appears, enter **d** to skip yet more special options.

   The space-time pattern is generated from the top down, on the left of the screen. To the right is a histogram of the lookup frequency relating to a window of 10 time-steps, and a plot of the entropy of this histogram, the input-entropy. An **on-the-fly key index** on the right of the screen gives a list of key presses to change settings on-the-fly. Try the following key presses to see what happens.

   - **#** to scroll the space-time pattern.
   - **g** to change the rule to different "complex" rule.
   - **u** to toggle the input-entropy - density plot.
   - **4** for a new random initial state.
   - **f** to progressively "filter" the space-time pattern, and **a** to restore the unfiltered pattern.
   - **e** and **c** to expand and contract the scale of the space-time pattern.

7. Try other key presses to change the rule, seed, color, analysis, size, updating, frozen cells, dimension, etc.

8. Enter **q** to interrupt and backtrack up the prompt sequence.

4.8 2d space-time patterns ("game-of-Life")

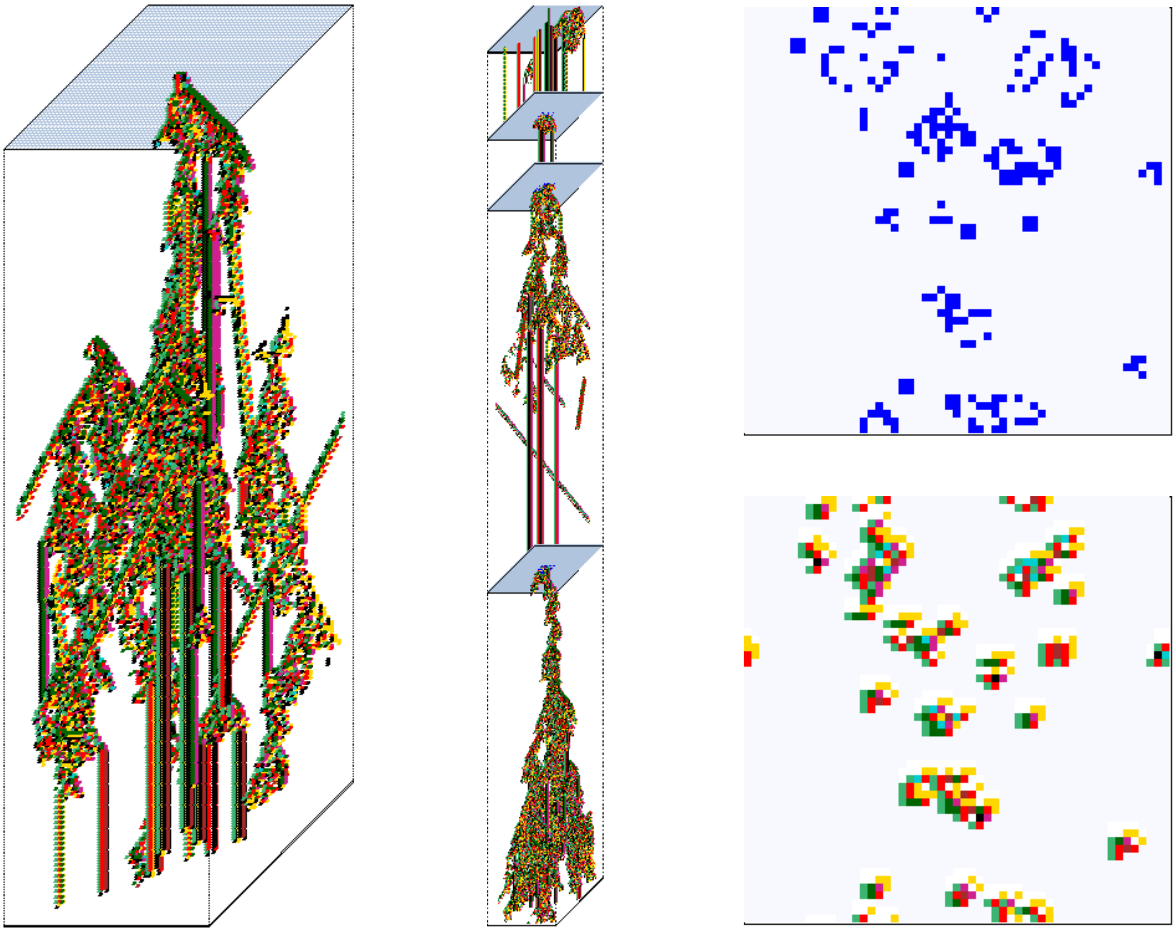


Figure 4.9: Space-time pattern of the 2d game-of-Life, (k=9, $n = 55 \times 55$) in a 3d isometric projection. 2d time-steps stack below each other, and are shown as if looking up at a transparent shaft. *left*: Starting from the "r-pentomino" seed. *center*: Re-scaled to the smallest scale, new seeds set at intervals. *lower right*: A particular state (time-step). *upper right*: A particular state colored according to the neighborhood look-up instead of the value.

Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt **forward only/single basin/subtree-s (default field):** enter **s**.

2. Enter **return** until a top right **WIRING** prompt window appears,

   WIRING: special-s load-l random-r
   regular: 3d-3 2d-2 1d-def: ... select 2 for regular 2d wiring.

   Any previous network and neighborhood size settings will be superseded.

3. Enter **return** until the top right prompt,

   2d, enter width (def-40): depth (def 40): set, say, 56×56. For a slow computer accept the default 40×40.

4. At the next top right prompt,
   Neighborhood size k: mixed-m, or enter 1-13 (def 3): enter 9.

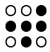
5. Enter **return** until the main sequence prompt for rule selection appears,

   Select k9-rule, empty-e fill-f maj-m Alt-a life-L chain-c rand-r
   bits-b hex-h repeat-p load-l (def-rand): select **L** for the "game-of-Life".

   The lookup-table of the rule will be displayed as a bit pattern.

6. Enter **return** until the **seed** prompt appears.

   Select SEED (2d ij=56,56), empty-e fill-f rand-r
   bits2d-b hex-h repeat-p load-l (def-r): enter **return** for a random block.

   Alternatively select **e** to "empty" all cells to zero, then **l** to load a seed, then enter the file name "pento" at the **LOAD SEED...** prompt. This is the "r-pentomino" pattern  that guarantees gliders.

7. Enter **return** until the top center **output parameters** banner appears, then enter **s** for **space-time pattern only**.

8. At the next prompt, **FORWARDS ONLY options:...**, a top center **accept defaults-d** banner appears, enter **d** to skip yet more special options.

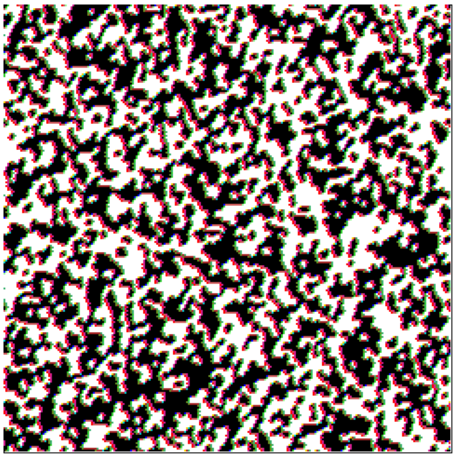
   The 2d space-time pattern is generated in the top left hand corner of the screen. The **on-the-fly key index** on the right of the screen gives a list of key presses to change settings on-the-fly. Try the following to see what happens.

   - **t** to toggle between the 2d display and a 3d isometric projection (imagine looking up at a transparent shaft).
   - **k** for a new random central block. **4** for a fully random seed.
   - **e** and **c** to expand and contract the scale of the space-time pattern.
   - **3** to toggle between cells colored according to rule-table lookup (the default) or by value.

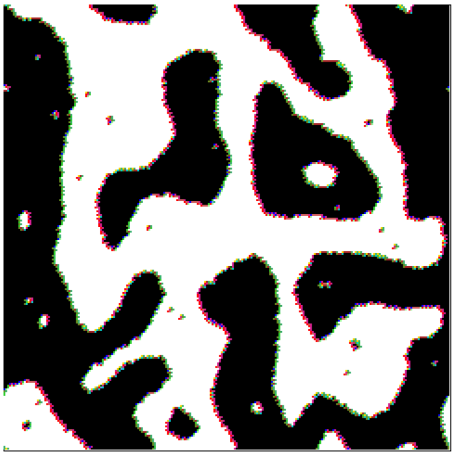
9. Try other key presses to change the seed, frozen cells, dimension, updating, rule, etc.

10. Enter **q** to interrupt and backtrack up the prompt sequence.

4.9 2d Space-time patterns (reaction-diffusion)



(a) k7 majority rule, totalistic code=11110000.



(b) k7 totalistic code=11101000.

Figure 4.10: 2d space-time patterns for a k7 CA on a triangular grid (240x240), from random initial states showing reaction-diffusion behavior.

Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt **forward only/single basin/subtree-s (default field):** enter **s**.

2. Enter **return** until a top right **WIRING** prompt window appears,

   WIRING: special-s load-l random-r
   regular: 3d-3 2d-2 1d-def: select 2 for regular 2-d wiring.

   Any previous network and neighborhood size settings will be superseded.

3. Enter **return** until the top right prompt,

   2d, enter width (def-40): depth (def 40): enter 240x240, or a smaller size for a slow computer.

4. At the next top right prompt,
   Neighborhood size k: mixed-m, or enter 1-13 (def 3): enter 7.

   This results in a triangular 2d array, where each cell has 6 nearest neighbors. Although shown orthogonally the underlying grid is effectively triangular.

5. At the **totalistic rule-t** prompt in the main sequence of prompts, enter **t**.

6. At the next prompt in the top right window enter "**return**".

7. Enter **return** until the main sequence prompt appears for selecting the rule as a totalistic code,

   Select k7 totalistic code (def-dec)
   maj-m rand-r bits-b hex-h repeat-p: select **b**.

   A flashing cursor will appear over a row of 8 gray squares. These represent the outputs of the 8 possible totals given 7 inputs, from 7-0, the totalistic code lookup-table. Set 1s in the table with key 1 (the square will turn red), set 0s with key 0 (the square with turn gray). Arrow keys move the cursor between squares. For space-time patterns as in figure 4.10, enter (a) 11110000, or (b) and 11101000.

8. Enter **return** until the **seed** prompt appears,

   Select SEED (2d ij=240,240), empty-e fill-f rand-r
   bits2d-b hex-h repeat-p load-l (def-r): select **r** for a random seed,
   then **a** for "all" the network, as opposed to a central block.

9. Enter **return** until the top center **output parameters** banner appears, then enter **s** for **space-time pattern only**.

10. At the next prompt, **FORWARDS ONLY options:...**, a top center **accept defaults-d** banner appears, enter **d** to skip yet more special options.

    The 2d space-time pattern is generated in the top left of the screen. An **on-the-fly key index** on the right of the screen gives a list of key presses to change settings on-the-fly. Try the following to see what happens.

    - **4** for a new random seed.
    - **e** and **c** to expand and contract the scale of the space-time pattern.
    - **3** to toggle between cells colored according to rule-table lookup (the default) or by value.

11. Try other key presses to change the updating, frozen cells, dimension, rule, etc.

12. Enter **q** to interrupt and backtrack up the prompt sequence.

4.10 3d space-time patterns

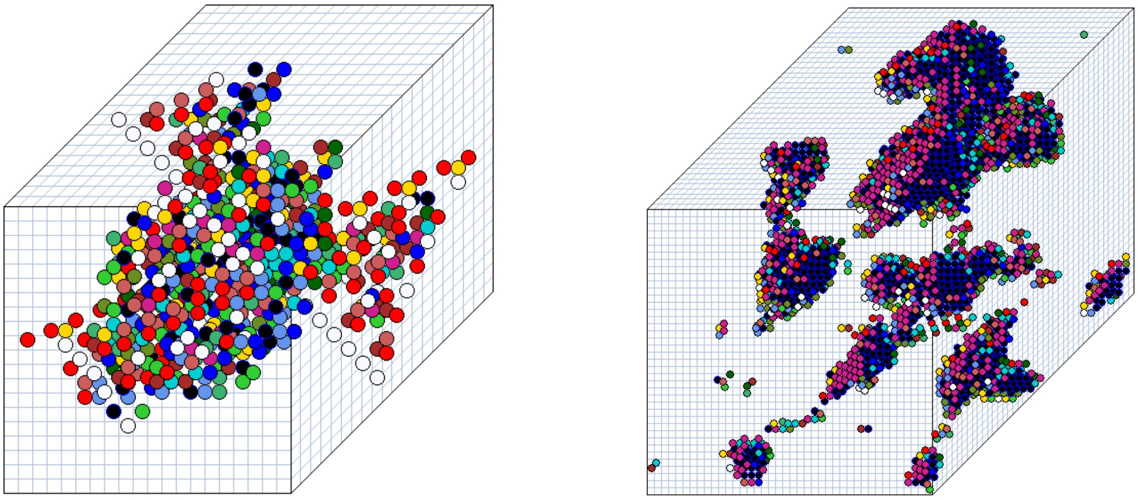
Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt **forward only/single basin/subtree-s (default field):** enter **s**.

2. Enter **return** until a top right **WIRING** prompt window appears,

   WIRING: special-s load-l random-r
   regular: 3d-3 2d-2 1d-def: select 3 for regular 3d wiring.

   Any previous network and neighborhood size settings will be superseded.



(a) $20 \times 20 \times 20$,
random $k7$ rule from a singleton seed

(b) $40 \times 40 \times 40$,
k7 totalistic code=11101000

Figure 4.11: Examples of 3d CA with a k=7 neighborhood arranged as a 3d cross. The projection is axonometric seen from below, as if looking up at the inside of a cage. Cells are shown colored according to neighborhood lookup by default for a clearer picture (instead of by value: 0,1).
(a) shows the evolution of a 3d CA, $k = 7$, $n = 20 \times 20 \times 20$, with a randomly selected rule. The initial state is a "singleton seed", a single *on* cell in an otherwise empty array.
(b) shows a 3d CA, $k = 7$, $n = 40 \times 40 \times 40$ (the maximum size DDLab supports), The initial state was set at random, but with a bias of 45% of *on* cells. The rule is the same as in section 4.9, showing reaction-diffusion behavior.

3. Enter **return** until the top right prompt, to sets the size of the 3d lattice,
   3d, enter width (def-9): depth (def 9): height (def 9): enter the width depth and height. The maximum cube would be $40 \times 40 \times 40$. The array has 3-torus boundary conditions.

4. At the next top right prompt,
   Neighborhood size k: mixed-m, or enter 1-13 (def 3): enter 7. The neighborhood is arranged as a 3d cross.

5. Enter **return** until the main sequence prompt for rule selection appears,

   Select k7-rule, empty-e fill-f maj-m Alt-a life-L chain-c rand-r
   bits-b hex-h repeat-p load-l (def-rand): enter **return** for a random rule.

   Alternatively follow the steps in section 4.9 to set the "totalistic code". The lookup-table of the rule will be displayed as a bit pattern.

6. Enter **return** (to accept defaults including the seed) until the top center **output parameters** banner appears, then enter **s** for **space-time pattern only**.

7. At the next prompt, **FORWARDS ONLY options:...**, a top center **accept defaults-d** banner appears, enter **d** to skip yet more special options.

The 3d space-time pattern is generated in the top left of the screen. An **on-the-fly key index** on the right of the screen gives a list of key presses to change settings on-the-fly. Try the following to see what happens.

- **5** for a new "singleton" seed, a single central 1, or **k** for a new random central 3d block.
- **r** for a new random rule.
- **e** and **c** to expand and contract the scale of the space-time pattern.
- **3** to toggle between cells colored according to rule-table lookup (the default) or by value.

8. Try other key presses to change the updating, frozen cells, dimension, etc.

9. Enter **q** to interrupt and backtrack up the prompt sequence.

4.11 "Screen-saver" demo

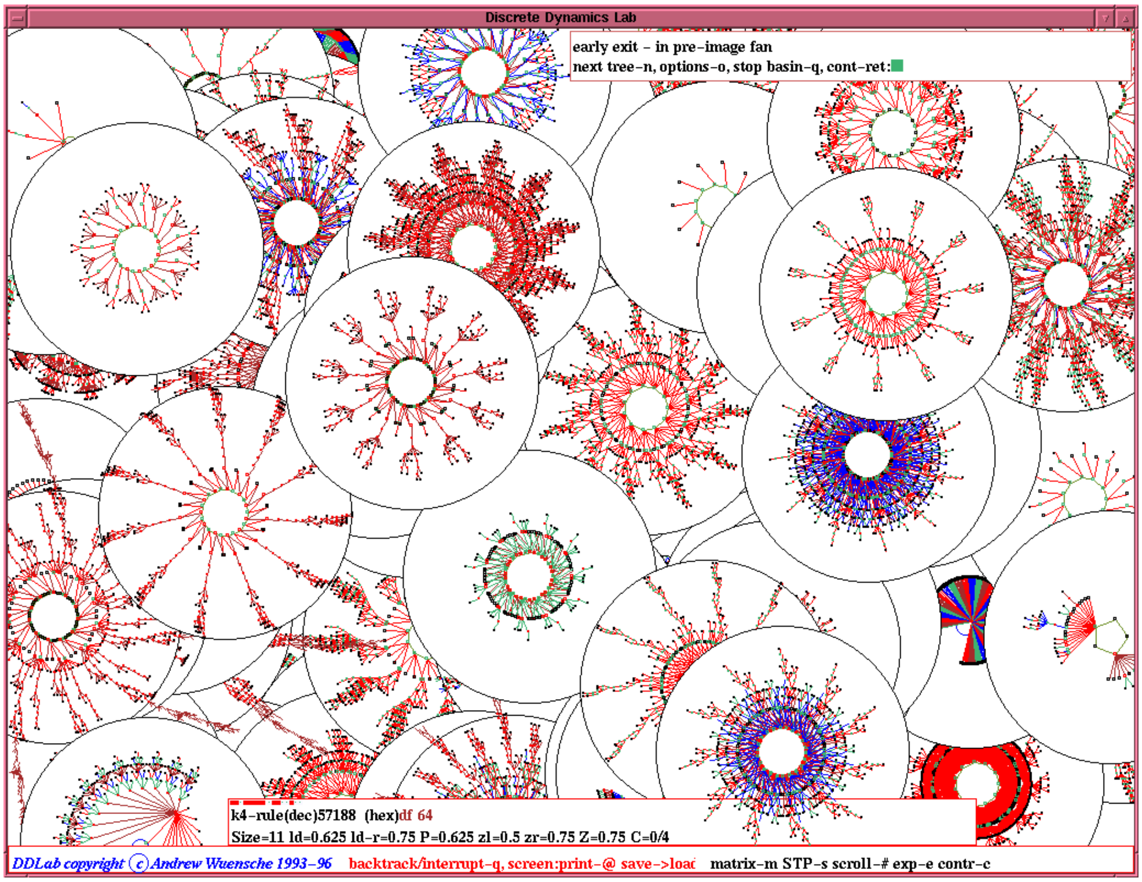


Figure 4.12: Screen-Saver demo for Cellular Automata, $n = 12$, $k = 4$. Single basins with rules and initial states chosen at random are generated at random positions in the DDLab window.

The "screen-saver" demo provides a continuous show of attractor basins "boiling" on the screen. Backtrack with **q** (or right mouse button) to the start of the program.

1. At the very first prompt **forward only/single basin/subtree-s (default field):** enter **s**.
2. At the **Network size...** prompt, select 12. (for a faster computer select 14).
3. At the **Neighborhood size...** prompt, select 4. Enter **return** until the top center **output parameters** banner appears, then **a** to restore all defaults, then **return** until the **savescreen demo -s** prompt; enter **s** to accept the option.
4. Then enter **d** twice (or **return**) to accept all further defaults. The demo will start. The rule picked at random will be shown in the bottom rule-window.
5. While the demo is in progress, toggle the display of the "backwards" space-time pattern with key **s**, and the state-space matrix with key **m** as described in section 4.3.
6. Enter **q** to interrupt and backtrack up the prompt sequence.

4.12 Random Boolean Networks

Until now, the Quick Start Examples have described cellular automata, which have a homogeneous *local wiring template* and a single *rule* throughout the network. A random Boolean network (RBN) departs from CA network architecture (the default in DDLab) by allowing each cell in the network to have different nonlocal *wiring*, a different *rule*, and different k (the number of inputs), or any combination of the above.

The network can be assigned a *wiring-scheme*, *rule-scheme* and *k-mix* in a variety of ways, including randomly (with or without biases).

- just a **wiring-scheme** may be set, the *rule* and k remain homogeneous, as in CA, by default.
- just a **rule-scheme** may be set, the *local wiring template* and k remain homogeneous, as in CA, by default.
- both a **wiring-scheme** and **rule-scheme** may be set, k remains homogeneous by default. This is what is usually understood as a "random Boolean network".
- a ***k*-mix** may be set, which implies a both *wiring-scheme* and *rule-scheme*, but the *wiring-scheme* remains *local* by default.
- a network with a ***k*-mix**, a non-local **wiring-scheme** and a **rule-scheme** may be set.

To generate attractor basins and space-time patterns for RBN, or any combination of parameters listed above, the preceding examples in sections 4.2-4.11 can be adapted as described below.

Note that because running RBN backwards imposes a greater computational load than for CA (see section 1.6), when generating attractor basins for RBN, both n and k should be kept small, and 2d or 3d networks should be avoided.

1. For a random **wiring-scheme** in 1d: At the first top right prompt,

   WIRING: special-s load-l random-r
   regular: 3d-3 2d-2 1d-def: select **r**.

2. For a random **wiring-scheme** in 2d or 3d: At the first top right prompt,

   WIRING: special-s load-l random-r
   regular: 3d-3 2d-2 1d-def: select **s**.

   At the next top right prompt **3d-3 2d-2 1d-def:** select **2** or **3**.

   At the next top right prompt, **hand wire-h,**
   regular: 3d-3 2d-2 1d-1, random-def: enter **return**.

3. For a random **rule-scheme**, at the top right prompt,

   RULES: single rule (def), load rule mix-l
   mix: no limit-n, or set limit up to 200: regular: enter **n**.

4. For a random **k-mix** for a 1d network, at the main sequence prompt, **Neighborhood size k: mixed-m, or enter 1-13 (def 3):** enter **m**. Various further options to bias the k-mix are presented. If in doubt enter **return**.

5. For a random **k-mix** for a 2d or 3d network, first select the a random **wiring-scheme** in 2d or 3d as above.

   At the top right prompt,
   Neighborhood size k: mixed-m, or enter 1-13 (def 3): enter **m**. Various further options to bias the **k-mix** are presented. If in doubt enter **return**.

6. To examine the resulting network in detail, at the the top right prompt (or similar),

   > **1d network (n=150),review/revise/learn, wiring and rules**
   > **graph-g, matrix: revise-m view-M**
   > **graphic: 1d+timestep-1 circle-c, 2d-2:**

   Enter **1**, **c**, **2** or **3** to review the network as a wiring graphic, using the arrow keys to examine the wiring and rules for chosen elements in the network. For 3d networks, the square bracket keys [and] move up and down in the 3d volume (see chapter 17).

   Enter **m** or **M** to review the network as a wiring matrix (see section 17.2)

   Enter **g** to review the network as a graph, which can be rearanged by dragging nodes with the mouse (see chapter 20).

   These options are also available for CA.

Chapter 5

Starting DDLab

This and the following chapters provide the detailed program reference, listing all the various options and prompts with explanatory notes. Before tackling this part of the manual, its a good idea to try the Quick Start Examples in chapter 4.

To start DDLab, make sure you are in the directory containing the DDLab executable file `ddlabx23` for Unix/Linux, or `ddlabx23.exe` for DOS. The files that were unpacked when DDLab was downloaded should also be in the directory. Chapter 3 gives downloading instructions.

5.1 Running in Unix/Linux

For Unix/Linux enter `ddlabx23 &` (or `./ddlabx23 &` on some Unix systems) , i.e. the name of the executable followed by `&` (to retain control of the xterm window). The DDLab window outline can be positioned on the screen with the mouse. Press **return** to start DDLab. The message `Using backing store` will appear in the xterm window indicating that the DDLab window will be restored in the usual way when covered by other windows or moved off the screen. If DDLab is iconised or resized, however, the current image in the window will be lost.

5.2 Running in a pure DOS environment

In a pure DOS environment, enter `ddlabx23` or the name of the executable without the `.exe` extension,. The `DOS extender` banner will briefly appear, and the program will start. To suppress this banner enter `set dos4g=quiet` at the DOS prompt before running DDLab. The resolution should be changed to correspond to the monitor and graphics driver, otherwise fonts might appear distorted (see sections 5.6 and 6.4.2 below).

To get into pure DOS from Windows95, from **Start** select **Shut Down...** then **Restart the computer in MS-DOS mode?**.

5.3 Running in Windows95

In Windows95, open a DOS window (from **Start** select **Programs** then **DOS Prompt**). Change to the `ddlabx23` directory and enter `ddlabx23`, or the name of the file without the `.exe` extension, as in section 5.2 above. DDLab will start either in a window or in full screen (which is unpredictable) but whichever it is **Alt** + **Enter** toggles between the two.

The initial resolution will be 640 × 480, anything higher may cause a crash so do not use the program parameter `-m` or `-h` (see sections 5.6 and 6.4.2 below). In full screen the resolution should be changed to correspond to the monitor and graphics driver (otherwise fonts may appear distorted). However, before toggling back from a full screen to a window, the resolution *must* be changed back to "low" to avoid Windows95 warnings and a possible crash. In higher resolutions the mouse cursor will probably not be visible.

DDLab is probably more stable in pure DOS mode (section 5.2 above) than in Microsoft Windows.

Figure 5.1: The graphical user interface when starting DDLab

5.4 Title bar

On start-up, DDLab's graphical user interface appears as described in section 4.1.

A title bar is displayed across the foot of the screen, including a copyright reminder, and some key press reminders which can be activated whenever the the green flashing cursor is showing. These are listed below,

DDLab copyright ©Andrew Wuensche 1993-2000 **backtrack/interrupt-q screen: print-@ save-> load-<**

key press	*what it means*
return ...	enter **return** (or click the left mouse button) to accept defaults to prompts and advance through the prompt sequence.
backtrack/interrupt-q ...	enter **q** (or click the right mouse button) to backtracks through prompts or to interrupt a run.
screen: print-@ ...	enter @ to print the DDLab screen image, described in section 5.5 below.
screen: save-> load-< ...	At any time when the green flashing cursor is showing, a screen image can be saved with < and loaded with < as a `.nat` file, DDLab's own format (see chapter 35).

Other context dependent reminders for key presses are displayed on the right of the title bar from time to time.

5.5 Printing the screen

If **@** is entered at any time when the prompt cursor is flashing, or if **p** is selected at various other print prompts in DDLab, the DDLab screen may be printed to the default printer.

5.5.1 Printing the screen in Unix

In the Unix/Linux version of DDLab, if the following prompt is presented,

print: without frame -p, with frame -P, for no greyscale add -n
then click mouse button in window, quit-q:

Press **p** to print the DDLab XWindow without its frame, **P** with its frame.

Add **n** (i.e. **pn** or **Pn** to print in black only, i.e. suppress greyscale or colors. With this option, some lighter colors will not be printed.

On pressing **return** a cross-hair (+) will appear. Make sure this is in the DDLab XWindow and press the left mouse button. You will hear a double blip. The DDLab XWindow will be saved automatically as a file called `temp_window_file.ps` (size about 285k). The file will be sent automatically to the default printer, which takes some seconds. While this is happening the message **wait** will appear in a top right window, followed by,

done - press any key to continue:

This reverts the program to the place before printing was selected. Note that the file `temp_window_file.ps` remains current until overwritten with another print command.

The following messages might be shown in the xterm window

```
pnmtops:  writing color PostScript...
xwdtopnm:  writing PPM file
pnmtops:  warning, image too large for page, rescaling to 0.7
```

The last indicates that the size of the DDLab XWindow was too large for the printer paper size, so was automatically rescaled to fit.

If the following error message is shown in the top right window,

could not print – xwdtopnm not found

or the following in the xterm window,

```
You need to set the environment variable ""XWDTOPNM_PATH""
to the full path of your xwdtopnm to print in greyscale
```

...consult you systems manager.

An alternative way to print the DDLab XWindow (or just part of it) is run XView (xv), then grab and save the image in one of the many file formats available, then print the file from the xterm window, for example `lpr -Pprintername myimage.ps`.

5.5.2 Printing the screen in DOS

In DOS, the DDLab screen can be printed at any time to the the Epson MX-82 dot matrix printer, or to the Cannon BJC 4000 bubble jet. These are the only printers that have been tested. Printing to other printers is worth a try but is unpredictable.

5.6 DDLab version and graphics info

A top right window shows the release date and current version of DDLab, information on graphics such as the screen/window resolution and text point size, the random number "seed" which changes each time DDLab is run, and a reminder that the left mouse button is equivalent to **return**, and the right mouse button is equivalent to **q** for backtracking through prompts.

Unix/Linux
DDLab v.23 July 2001 window=1024x768 pt=14 randseed=48639
mouse buttons: left=ret right='q'=backtrack

DOS *(screen resolution is 640×480, 800×600 or 1024×768)*
DDLab v.23 July 2001 screen=640x480, randseed=11291
mouse found: buttons: left=ret right='q'=backtrack *(if a mouse is detected)*
or
mouse not found *(probably because of a missing DOS mouse driver)*

For DOS the default screen resolution is 640×480 pixels (VGA). With the appropriate monitor and graphics driver, the program parameter `-m` or `-h` will start DDLab at a higher resolution, 800×600 or 1024×768 pixels (SVGA). The resolution can also be changed within DDLab, see section 6.4.2 for details.

For Unix/LINUX the default DDLab window resolution is automatically set to fit comfortably on the monitor screen. The window can be resized to an exact size (see section 6.4), or with the mouse in the usual way, and the text size will attempt to resize accordingly.

5.7 The mouse cursor in DOS

On newer systems, especialy laptops running Windows98, the mouse in DOS mode may not be supported. The message **mouse not found** may be displayed in section 5.6. For the mouse cursor to work, a seperate mouse driver for DOS may need to be installed. In Windows98, the mouse may be recognized but not visible. This is an unresolved problem.

5.8 Memory (DOS only)

DDLab allocates and frees memory as required while the program runs. In DOS mode the memory available is monitored, mainly for diagnostic purposes.

5.8.1 Memory available

A small top center window shows the amount of RAM available in bytes (including extended memory), for example **mem=4146056**. This is frequently updated. Other information is also displayed in this window from time to time.

5.8.2 Virtual Memory

You may create a swap file on disk to augment RAM, using Watcom's "Virtual Memory Manager" (VMM), and use more memory than your computer actually has. When RAM is not sufficient, part of the program is swapped to the disc file until needed. The combination of the swap file and available RAM is the virtual memory.

To specify virtual memory, set the DOS environment variable, "dos4gvm" by entering the following at the DOS prompt:

`set dos4gvm=[option[#value]] [option[#value]]...`
or
`set dos4gvm=1` *to accept all defaults*

the VMM options and default values are listed below,

`minmem` The minimum amount of RAM managed by VMM, default 512KB.
`maxmem` The maximum amount of RAM managed by VMM, default 4MB.
`swapname` The swap file name, default `dos4gvm.swp` in the root directory.
`virtualsize` The size of the virtual memory space, default 16MB.

For example, typing

```
set dos4gvm=maxmem#8192 virtualsize#32768 swapname#c:\ddlab\ddlab.swp
```

at the DOS prompt before running DDLab, gives 32MB of virtual memory in a swap file called `ddlab.swp` in the `ddlab` directory. The virtual memory available will be displayed in the memory window, as described in section 5.8.1 above.

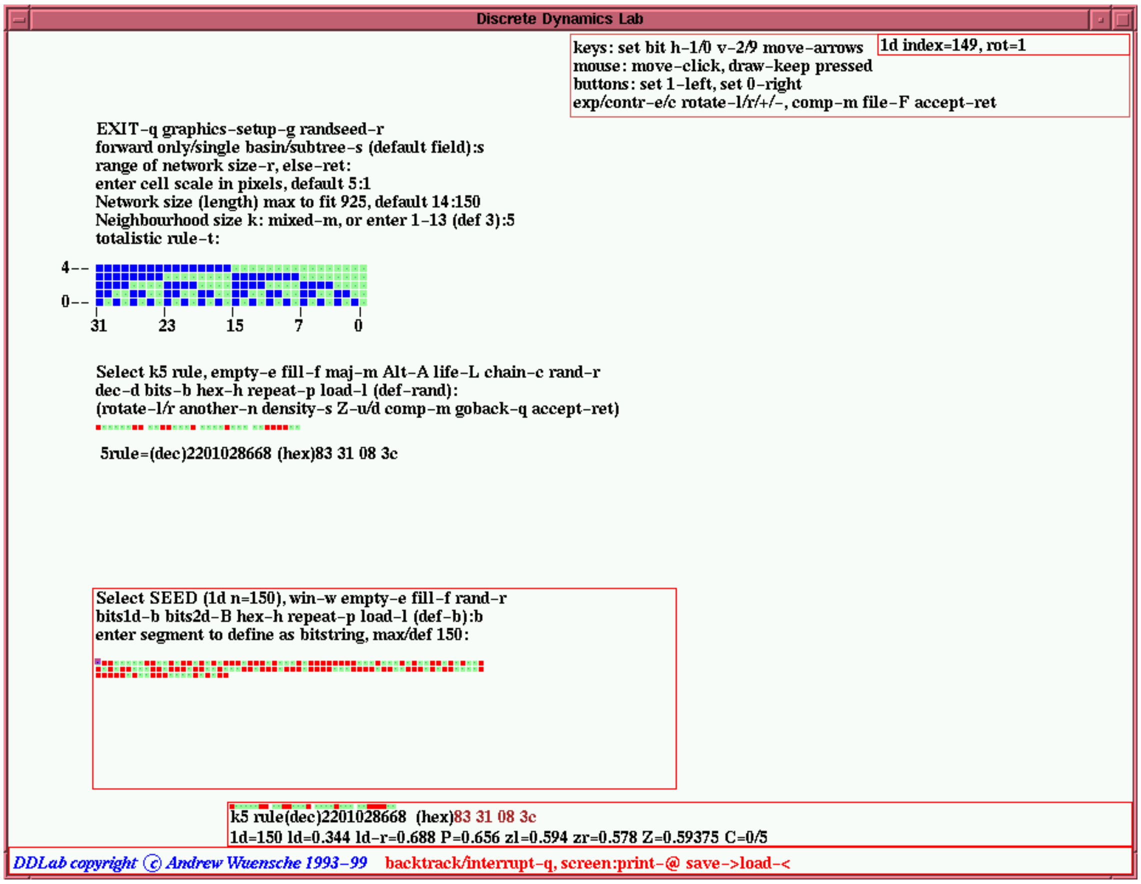


Figure 5.2: The main sequence of prompts, for the most commonly applicable 1d CA parameters, are presented down the left side of the screen, other prompts occur in various pop-up windows, for example in the top right hand corner.

5.9 DDLab prompts

The first of a series of prompts for setting network and presentation parameters is displayed, described in chapter 6.1. A flashing cursor (usually green) prompts for input. Just enter **return** if in doubt, or the appropriate input from the keyboard. Press **q**, **back-space**, (or the right mouse button) to revise. **return** (or the left mouse button) to accept and move on to the next prompt or routine. Just **return** (or left mouse button) automatically selects a default. To backtrack to the preceding prompt, revise, or interrupt a running routine such as space-time patterns or attractor basin, press **q**, or right mouse button. To quit DDLab immediately (not for DOS) enter **Ctlr-q** at any prompt, followed by **q**. Otherwise backtrack with **q** to the start of the program.

Prompts occur in a main sequence for the most common 1d CA parameters, and also in various pop-up windows for RBN, 2d and 3d networks, analytical measures, data collection, and presentation setting. Note that for 2d and 3d CA and RBN, the main sequence prompts for the network and neighborhood size are superseeded in the first pop-up window, labeled "WIRING".

Chapter 6

The first prompt in DDLab

This chapter describes the first prompt in DDLab, which determines if an initial state will be required for the simulation. There are also various options for graphics, filing, printing and the random number seed.

6.1 The first prompt

The first prompt in DDLab is as follows,

> **EXIT-q graphics-setup-g randseed-r**
> **forward only/single basin/subtree-s (default field):**

6.2 Field, or a run requiring a seed

A choice needs to be made at the start whether an initial state (the *seed*) will be required for the simulation. A seed is needed to run a network forward for just space-time patterns, and also for single basins of attraction or sub-trees.

Generating a "basin of attraction field" (the default), does not require a seed because successive basins are seeded automatically, starting with the null state (all 0's).

Select **s** at the first prompt in section 6.1 for a **forward only** run (i.e. to generate just space-time patterns) or for a **single basin** or **subtree**. This tells DDLab that a seed will be required, which will be set at a later stage in the prompt sequence (see chapter 15).

Enter **return** for a basin of attraction field.

6.3 Graphics setup

Enter **g** in section 6.1 to change the window size, background color, font size, line spacing or flashing cursor speed. A new graphics screen shows the current font and color palette. A top right window gives prompts as follows,

Unix/Linux
window size=925x694 resize-S *(defaut resolution depends on monitor)*
character width=9 height=15 pointsize=14 resize-s *(example text character info)*
colors: toggle background-b showold-o showall-a
flashing cursor speed-f, print-p quit-q cont-ret:

DOS
screen=640x480 change:1024x768-h 800x600-m *(where h=high m=medium l=low)*
colors: tog background-b
fontsize-s linespace-v
flashing cursor speed-f, quit-q cont-ret:

6.4 Screen/window resolution

In Unix/Linux the resolution of the initial DDLab XWindow is set automatically to fit comfortably on the monitor screen. On my laptop the default resolution is 925x694, but can be resized with the mouse in the usual way, or a particular size can be set from prompts.

In DOS or Windows95 there are three alternative resolutions, given an appropriate monitor and graphics driver.

6.4.1 The DDLab XWindow resolution, Unix/Linux

Enter **S** in section 6.3 to resize the DDLab window to a particular size in pixels.

The following top right prompt is presented,

new window size (min 300x250)
width (default 925): *(for example)*
height (default 694):

The smallest size allowed by this method is 300×250 shown in figure 6.1. The window can also be resized to any size with the mouse by dragging a corner in the usual way. Note that when the window is resized the font size is automatically changed to an appropriate size. However, a very small size will inevitably produce unforeseen effects.

6.4.2 The DDLab screen/window in DOS and Windows 4.1/95/98

The DOS version of DDLab can be run as a DOS application in Windows95 (in a small window or full screen), or in a pure DOS environment, which is recomended.

DDLab will start in low resolution (640×480) whatever the screen resolution, unless the program parameter -m or -h is entered, i.e. ddlabx22 -h. If this is done without the appropriate monitor and graphics driver the results are unpredictable and may cause a crash. If running in a DOS window start in low resolution.

Select **-l**, **-m** or **-h** in section 6.3 above to change the resolution.

l=low:640x480, **m**=medium:800x600 and **h**=high:1024x768 resolution.

If running DDLab from a DOS window in Microsoft Windows 4.1/95/98, **Alt** + **Enter** toggles between a full screen and a window. In a small DOS window, only low resolution is permissible. In full screen the resolution may be changed to medium or high. However, before toggling back from a full screen to a window, the resolution *must* be changed back to low to avoid Windows warnings and a possible crash. DDLab is probably more stable in pure DOS mode (see section 5.2) than in Microsoft Windows.

6.4.3 White or black background

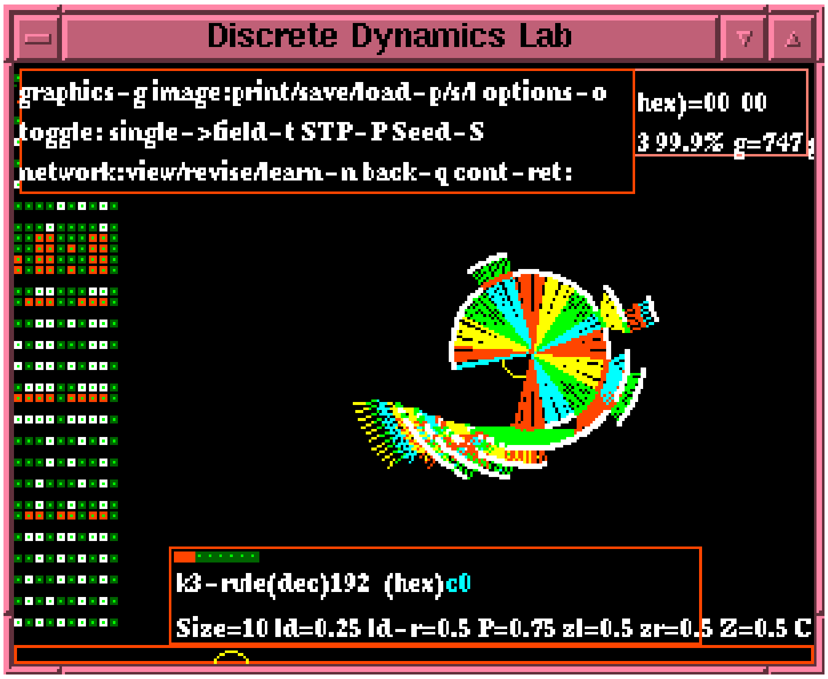


Figure 6.1: The smallest DDLab window in Unix/Linux that can be set from the graphics setup prompts. It is really too small to contain prompts and data. This example has the default black background.

Select **b** in section 6.3 to toggle between a black and white background. In both cases the new color palette will be shown. A black background may look more elegant but is expensive in ink when printing. When the graphics changes are complete the program reverts to the first prompt in section 6.1. References to colors in this manual are based on a white background. Colors on a black background may be different. In DOS, toggling the background works in full screen, not in a DOS window.

6.4.4 The color palette

The graphics screen shows a color palette of the 16 colors (including black and white). In DOS just these colors are used. In UNIX these are the main colors, but more colors are also used, and can be seen by entering **a** (**showall-a**) in section 6.3, reverting to the 16 main colors with **o** (**showold-o**. This feature is intended mainly for program development.

6.5 Changing the font size and line spacing

The default font size and spacing between lines of text can both be changed. The methods differ between Unix/Linux and DOS.

6.5.1 Font size and line spacing, Unix/Linux

If **s** is entered in section 6.3, the following top right prompts are presented in sequence,

set new font size (min 7 max 35, now 14:
set new linespacing (now 15): *(values shown are examples)*

Enter the new font size, followed by the new spacing.

6.5.2 Font size, DOS

If **s** is entered in section 6.3, a new screen appears showing some text in numbered fonts of various sizes, including a line segment font, and the following prompt,

font 5, to change enter font no: *(values shown are examples)*

Enter the font number corresponding to the new font.

6.5.3 Line spacing, DOS

If **v** is entered in section 6.3, the following top right prompt is presented,

set new linespace (now 20): *(values shown are examples)*

Enter the new line spacing.

6.6 Changing the flashing cursor speed

The speed at at which cursor flashes depends on the speed of your CPU and other factors. However the flash rate can be altered, enter **f** in section 6.3. The following top right prompts are presented in sequence,

change flashing cursor: restore-r or enter factor:

Enter a factor to slow down or speed up the flash rate. For example to make it half as fast enter .5, twice as fast enter 2. To restore the default setting enter **r**.

6.7 Random number seed

A new random number seed is generated by the computer's "timer" whenever DDLab is started for the first time. To change the random number seed, enter **r** at the first prompt in section 6.1. A specific random number seed can be set, or a *random* random number seed . The following prompt appears in a top right window,

enter random number seed 0-32767 (default random):

Using the same random number seed results in the same sequence of pseudo random numbers being produced by the random number generator. This allows identical multiple runs using default random settings of network parameters such as wiring, neighborhood mix, rules and initial state. Alternatively these setting can be loaded from previously saved files.

6.8 Exit DDLab

Figure 6.2: Warning prompt to exit DDLab, or restart.

To exit DDLab enter **q** in section 6.1, or **Ctrl-q** at any later prompt (not for DOS). An exit warning is displayed at the center of the screen, figure 6.2. Enter **q** to exit DDLab (or **return** to restart)

Exiting in Unix/Linux destroys the DDLab XWindow, and **DDLab clean exit** should appear in the xterm window. In DOS/Windows the system returns to the DOS prompt,

Chapter 7

Network size, 1d

This chapter describes how to set the 1d network size (or range of sizes), and also the "cell scale", the size of the representation of a network cell. Limits to the network size are also discussed.

Note that to specify a 2d or 3d network and set its size, these 1d prompts can be ignored by entering return.

7.1 Setting range of sizes, 1d

It is possible to generate a series of basin of attraction fields for a range of 1d network sizes (see figure 7.1), with the network size printed on the left of the screen. The presentation is similar to the "Atlas" in "The Global Dynamics of Cellular Automata" [12]. Single basins or subtrees for a range of network sizes may also be generated.

The following main sequence prompt is presented,

range of network size-r, else-ret:

If **r** is selected the next two prompts are as follows:

Start size, default 5:
End size, default 12:

Accept the defaults, or enter the start and end sizes, taking care to avoid excessively large sizes (see section 7.4 below).

7.2 Setting the size of one network, 1d

If **r** was not selected in section 7.1 above, the 1d network size is set with one of the following main sequence prompts (the various values are examples),

Network size (length) max 31, default 10: *(for a* **field***, selected in section 6.1)*
Network size (length) max to fit 170, default 14: *(if* s *was selected in section 6.1)*

If **s** was selected at the first prompt in section 6.1, a prompt to select the cell scale will be presented before this prompt (see section 7.3 below).

For 2d or 3d networks, the network size will be set as part of the wiring options (chapters 10 - 11), so the prompts in sections 7.1- 7.2 may be ignored by pressing return.

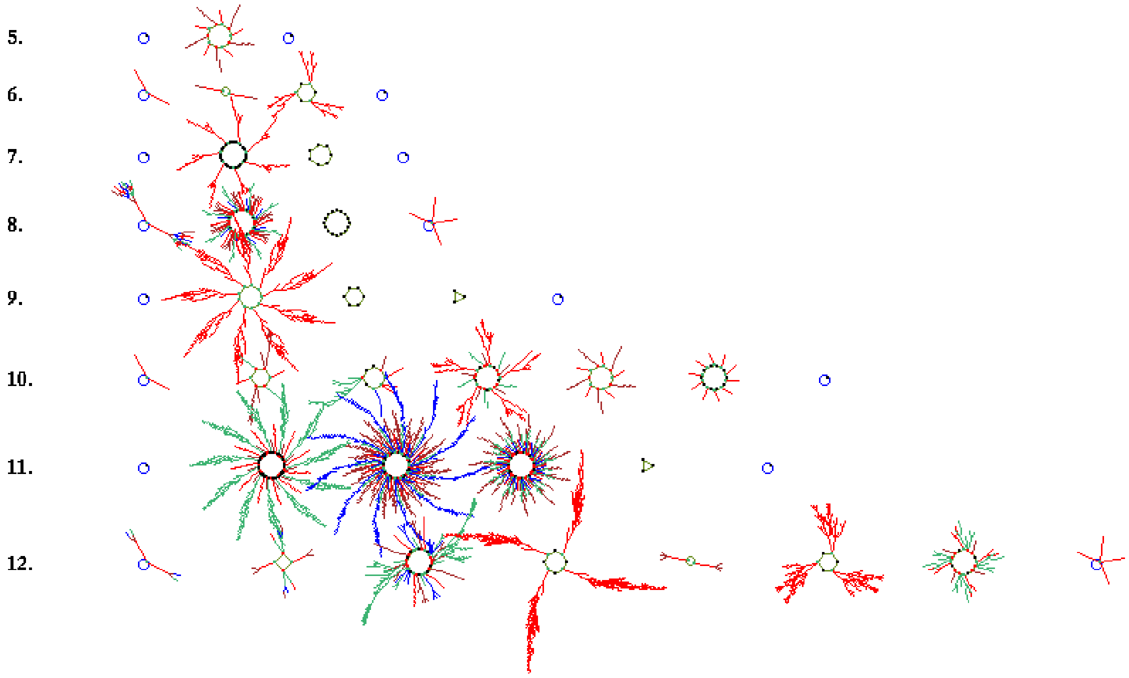


Figure 7.1: Basin of attraction fields for a range of network size 5-12. $k = 4$, rule=93cc

7.3 Cell scale

This defines the size of the graphic representation of a network cell in pixels, setting the scale of a cell for space-time patterns, and also for bit patterns at attractor basin nodes.

When running "forward only" with a system size of say 150 set this to 1, otherwise accept the default. The cell scale can be changed at other points in the program and also on-the-fly.

The following prompt is presented,

enter cell scale in pixels, default 4:

The default depends on screen/window resolution, for example for 925x694 the cell scale = 5 pixels).

7.4 Network size limitations

Network size limitation are also discussed in section 1.6. To summarize, the maximum network size, n_{max}, supported by DDLab is as follows,

- **Space-time patterns (1d, 2d or 3d):** n_{max}=65025, which corresponds to the maximum size of a square 2d network, 255×255, and can be represented by an unsigned short (16 bits). The maximum size of a 3d cube is 40×40×40.

  For 1d networks, a much smaller size is preferable, say n=150 to 200, to fit the screen and for a clear view of analytical graphic windows.

- **Sub-trees:** n_{max}=65025. In practice the size should be much smaller, say n_{max}=50. For a practical size, try n=25.
- **Single basins:** n_{max}=65025. In practice the size should be smaller than for sub-trees, say n_{max}=25. For a practical size, try n=18.
- **Basin of attraction fields**[1]**:** n_{max}=31. In practice n_{max} should be much smaller, say n=20. For a practical size, try n=16.

In general, large networks sizes may be run "forwards" to generate space-time patterns, but only very small networks should be run "backwards" to generate attractor basins. Of these, a subtree allows the largest networks, and basin of attraction fields the smallest. If in doubt, small sizes should be tried first. Larger sizes might exhaust computer memory or take an excessive length of time to compute, especially for networks other than regular 1d CA.

There is no hard and fast rule for deciding what size is too large for attractor basins. For CA it depends on the rule itself and its Z parameter[19]. If a rule is "chaotic", with high Z and a low "in-degree" (low branchiness), larger sizes may work. Ordered rules, with low Z and high "in-degree" require the smallest sizes.

The size limitations also depend on the degree of connectivity k, especially for RBN. Greater k results in slower computation.

7.5 Times for basin of attraction fields

The times needed to generate basin of attraction fields for a range of network sizes, for both CA and RBN, and for two generations of PC, are set out below (the time is displayed in DDLab).

The system size that is appropriate for generating attractor basins in a reasonable time can be inferred from these examples. Large sizes may be tried, but may impose unacceptable time, memory and display constraints. Single basins, and especially sub-trees, may be generated for relatively much larger systems, especially for rules having a low in-degree.

7.5.1 Examples for CA

Examples of the time needed to generate the basin of attraction field of some 1d CA rules for a range of k and n, is shown in table 7.2. With some notable exceptions, the time doubles with increasing n, and by a factor of about 1.6 with increasing k,

7.5.2 Examples for RBN

A similar exercise for RBN (wiring and rules selected at random) gave the more erratic timings shown in table 7.2, especially sensitive to k. The time required depends on n, k, the particular wiring and rule mix, and the resulting topology of attractor basins (the greater the characteristic in-degree, the longer the time).

[1] This size limit allows an unsigned long to be the index to bits in a char array that identifies "used" states.

n	12	13	14	15	16	17	18	k and rule
time	2s	4.9s	8s	16s	32s	1m 4s	2m 13s	k=3 rule (dec) 193
	3.8s	7s	13s	27s	50s	1m 47s	3m 37s	k=4 rule (hex) e9 24
	5.4s	9.8s	30s	37s	1m18s	2m 36s	8m 46s	k=5 rule (hex) b7 55 d3 d9

The DOS version of DDLab on a 66MHz 486 PC.

n	12	13	14	15	16	17	18	k and rule
time	0.5s	1.0s	1.5s	2.7s	5.7s	10.7s	22.3s	k=3 rule (dec) 193
	0.8s	1.3s	2.3s	4.8s	8.1s	17.7s	34.4s	k=4 rule (hex) e9 24
	0.9s	1.6s	4.7s	5.7s	11.8s	23.4s	1m 18s	k=5 rule (hex) b7 55 d3 d9

The Linux version of DDLab on a 233MHz Pentium PC.

Table 7.1: Time measures for basin of attraction fields - CA.

n	12	13	14	15	16	k
time	12s	21s	40s	1m 13s	3m 21s	2
	18s	50s	1m8s	2m 43s	8m 39ss	3
	60s	2m8s	4m33s	34m 32s	87m 20s	4

The DOS version of DDLab on a 66MHz 486 PC.

n	12	13	14	15	16	k
time	1.9s	8.0s	12.8s	15.6s	1m 0s	2
	8.0s	18.4s	26.3s	50.6s	2m 11ss	3
	10.5s	32.1s	1m 44ss	4m 31s	8m 54s	4

The Linux version of DDLab on a 233MHz Pentium PC.

Table 7.2: Time measures for basin of attraction fields - RBN.

Chapter 8

Neighborhood, k, or mixed k

A homogenious neighborhood size, k, may be specified for the whole network. Alternatively, a mix of k sizes may be selected, with the exact "k-mix" to be specified subsequently.

8.1 Maximum and minimum k

The maximum number of input wires to a cell, k_{max}, currently supported in DDLab is 13, and the minimum is 1. k=0 is illegal as it would require an array size of zero, but an effective k=0 can be easily set as described below.

8.2 Effective k=0

Setting k=0 directly is illegal in DDLab. To set an effective k=0, set k=1 with the cell wired to itself (local wiring in 1d, 2d or 3d), and set its rule to (dec) 2, or 10 in the rule-table. This ensures that the cell is not recieving any inputs from other cells, and that it will conserve its current value. However, other cells may recieve inputs from the k=0 cell.

Setting k=0 is mainly usefull in mixed k networks. DDLab can identify any self-wired k=1 cells, and gives an option to set the rule at these cells to make them effectivly k=0 (see section 13.2).

8.3 Selecting k, or a k-mix, for 1d networks

To select k, or a k-mix, the following prompt is presented, either in the main sequence of prompts for 1d networks, or in a top right wiring window for 2d or 3d networks,

Neighborhood size k: mixed-m, or enter 1-13 (def 3):

Enter enter the required value of k, or **m** for a k-mix. A k-mix is set in section 8.4 below. For 2d and 3d networks, both the mix and size will be reset in the wiring options (chapters 10 - 11), so the main sequence prompt can be ignored by pressing **return**). The maxumum maximum k set becomes the new default in this option.

Once the network's rule or rules have been set (chapters 12-16), the network's k or k-mix may be "neutrally" modified, both by increasing k or reducing to "effective k", for the network as a whole or for particular cell.

8.4 Specifying the k-mix

If **m** is selected in section 8 above, the following prompt appears in a top right window,

set k-mix: load-l hand-h norm-n power/specify-s rnd-(def):

The k-mix can be set in a variety of ways described below.

Note that settings for particular cells in a k-mix can be changed later in the "wiring graphic" options (see chapter 17).

8.5 Loading a k-mix file

Enter **l** is selected in section 8.4 above to load the k-mix as a `.mix` file (see Filing, chapter 35). The file would have been saved in sections 19.4 or 8.12.3. If the file is too big (contains excess k-mix data), as much k-mix data as will fit into the network will be loaded, starting from cell index 0. If on the other hand the file is too small (contains insufficient k-mix data), all the data will be loaded from cell index 0, the excess cell indices will be allocated a uniform background mix with the following prompt,

k-mix loaded (10) less than net size (150) *(values shown are examples)*
enter background k-mix 1-13, (default 5):

The k-mix encoding is described in section 19.7.1. The k-mix can be also be save/loaded in section 19.1 for network architecture, (see also section 17.8.12).

8.6 Setting the k-mix by hand

If **h** is selected in section 8.4, a series of prompts allow the k at each cell index (starting from 0) to be set.

set neighborhood size (1-13) at cell 0 (back-b):

Enter the required k, or **return** for a random k value between 1 and 9. **b** allows backtracking to the previous cell index.

8.7 Setting a normal distribution

Enter **n** to set a normal (Poisson) distribution of the frequencies of different k. This is achieved by assigning potential links at random to the network, up to a preset limit. However, every cell will be assigned at least one link. The following top right prompts are presented in sequence,

set k upper bounds (def 13, Max 13):
set average k (def 2.60, max 8.67): *(values depend on k upper bounds)*

Enter the upper bounds of k for the distribution, followed by the target average k in the network. An example of a distribution is shown in figure 8.

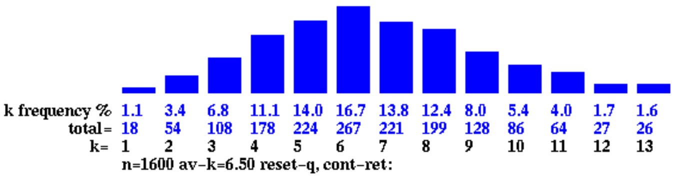


Figure 8.1: An example of a normal (Poisson) distribution of the frequencies of different k. n=40×40, the range of k=13 and the average k=6.5

8.8 Specify each k, or power-law distribution

Enter **s** in section 8.4, to specify the distribution of different k, or to select a power distribution. The following top right prompt is presented,

power-law-p, specify (def):

8.8.1 Specify the percentage of different k

If the **return** is entered in section 8.8 above, the percentage of cells with different k can be specified, and within those constraints the k-mix is assigned at random (i.e. shuffled). Alternativly, the k-mix need not be shuffled in which case it is assigned to the network in continuous blocks.

The following series of prompts are presented for each k starting with k=1 (shown with example settings),

enter % k=1 (100.0% left):22 *(for 22% k=1)*
enter % k=2 (78.0% left) back-b:25 *(78.0% of the network remains)*
enter % k=3 (53.0% left) back-b:33 *(53.0% of the network remains)*
...etc.

The prompts continue until none of the network remains, or until the k=13 prompt. **Return** does not assign any of that particular k. An assignment that is too large for the network still remaining will be truncated. If on completion some of the network remains, this will be set with the last k that was assigned. If none of the k's are assigned, the prompt reverts to section 8.4. At any stage it is possible to backtrack and revise with **b**.

8.8.2 Setting a power-law distribution of k

Enter **p** at the prompt in section 8.8 to set a power-law distribution of the frequencies of different k. However, every cell will be assigned at least one input. The following top right prompts are presented in sequence,

set k upper bounds (def 13, Max 13):
enter power-law exponent 1 to 3 (def 2.0):

Enter the upper bounds of k for the distribution, followed by the taret average k in the network.

An example of a power-law distribution is shown in figure 8.8.2. The output distribution can also be set to approximate a power-law (see section 17.8.4). If both the inputs, k, and the outputs, follow a power law, the network is said to be "scale-free". This is supposed to be characteristic of many natural and artificial networks, from the internet to metabolic networks [1]. The graph of such a network is shown in figure 20.

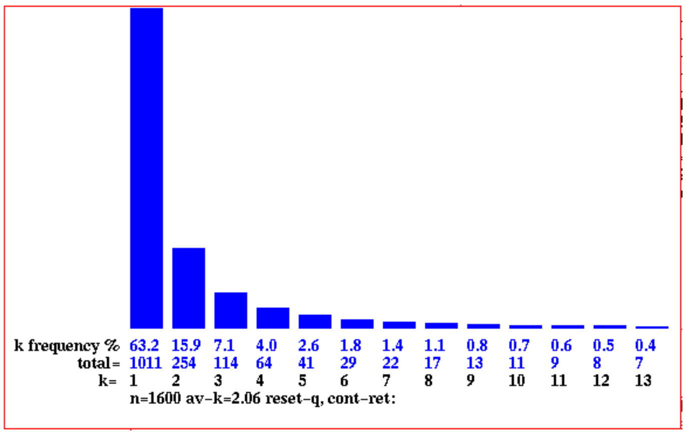


Figure 8.2: An example of a power-law distribution of the frequencies of different k. In this case n=40×40, the range of k=13, the power-law exponent is 2, and the average k=2.05

8.8.3 Showing the distribution

When distribution of different k's have been assigned, either by specifying in section 8.8.1, or by setting a power-law in section 8.8.2, data on the distribution, and the following prompt, are presented in a top right window (for example, for a 2d 40×40 network, and a power-law),

k1=63.2%=1011 k2=15.6%=245 k3=7.15%=114 k4=4.0%=64 k5=2.6%=41
k6=1.8%=29 k7=1.4%=22 k8=1.1%=17 k9=0.8%=13 k10=0.7%=11
k11=0.6%=9 k12=0.5%=8 k13=0.4%=7
shuffle: no-n yes-(def):

This shows the percentage and number of different k's in the network. The k's are initially assigned in continuous blocks starting from the network cell 0. Enter **return** to randomly shuffle the assignment, or **n** *not* to shuffle and retain the blocks.

The distribution can also be displayed as a histogram (see section 8.12, as show in in figures 8 and 8.8.2.

8.9 Setting the k-mix at random

If **return** was entered in section 8.4, the default, a random k-mix is selected. A further prompt allows the k-mix be confined within upper and lower bounds before the random mix is assigned. The different k's will be assigned equaly as far as possible to the network,

set k bounds 1-13: lower (def 1): upper (def 9, Max 13):

8.10 Setting a k-mix with uniform k

Its sometimes usefull to set up a k-mix'd network with homogeneous k, for example to set up a regular 1d, 2d or 3d CA into which a random Boolean sub-network can be inserted (see section 19.8). Sub-networks, possibly with mixed k, provide a souce of noise in the regular network.

A k-mix'd network with homogeneous k is easily created in section 8.8.1, by setting the percentage of the required k at 100%, or in section 8.9 by setting equal values for the lower and upper bounds in a "random" the k-mix. Such a homogeneous k network is still treated as a mixed k network, and as such is also treated as a having non-local wiring and mixed rules even if the wiring is set as local 1d (see section 11.4.1), and there is just one rule (see section 12.6). These issues relate to how memory is allocated to different types of network.

8.11 Increasing max-k

Options for changing k for individual cells (or blocks of cells) occur at later stages in DDLab (see sections 17.8.7 and 18.9). However, k is limited to a maximum value, max-k, because of the way network memory is allocated. The default max-k is the maximum k found in the network. However max-k can be set to a higher value (up 13, the maximum k currently supported by DDLab). The following top right prompt is presented,

set greater max-k (def 8): *(if 8 was the max k found in the network)*

This enables the k values in the network to be subsequenty increased up the new max-k.

For example, if 100% k=5 is set in section 8.8.1 (a "k-mix" with uniform k), and max-k is set to say 9, cells in the uniform k network may later be reset to any value ≤ 9.

8.12 Reviewing the k-mix

Finally, the k-mix is displayed in a window on the right of the screen, or a succession of windows for large networks. For example, figure 8.3 shows a randomly assigned k-mix between 1 and 13 to a network size 200, for each cell, 199 to 0. The k assignments are shown in hex. k's 10-13 are **a-d** shown in color. The percentage and number of different k's in the network are also displayed.

Options are presented at the foot of the k-mix review window,

```
k-mix 1-13, maxk=13, net size=200, startindex=199
498556c514537266446b7b119551bb3d1d5a2d4dc29dbd8b5ac7866819587d7
ac2d8b2455a518c1b26325b47481b16d55b46a86492ac9a216bb9a877abb121
2a865758b7a8645b35249c9ba3278c611bac831954b39d76872694b416b9735
7cbd2426946
1=8.5% 2=8.0% 3=4.0% 4=8.5% 5=10.5% 6=9.5% 7=7.5% 8=8.0%
9=7.0% 10=6.5% 11=11.5% 12=5.0% 13=5.5%
n=200 tot_wires=1365 av-k=6.82, mix shown from cell index 199-0
hist-h reset-r save-s jump-j cont-ret:
```

Figure 8.3: An example k-mix, randomly assigned for k=1-13, for a network size=200. The k assignments are shown in hex.

...
mix shown from cell index 244-0 *(values shown are examples)*
hist-h reset-r save-s jump-j cont-ret:

Enter **h** to show the k distribution histogram in a lower left window, as in figures 8 and 8.8.2. Enter **r** to revert to section 8.4 and reset the k-mix. Enter **s** to save the histogram data to a `.his` file (see Filing, chapter 35); the data will also be printed in the xterm window (*not for DOS*). Enter **return** to accept the k-mix. The other options are explained below.

8.12.1 Reviewing the k-mix in large networks

Large networks may require several windows to display the k-mix. In this case the prompt above (section 8.12) includes an option **m** to see successive windows (for example),

...
mix shown from cell index 7011-4221 *(values shown are examples)*
hist-h reset-r save-s more-m jump-j cont-ret:

8.12.2 Jumping to a new cell index

Alternatively, enter **j** at 8.12.1 to jump to any cell index in the network, which becomes the first index in the new window. The following prompt is displayed,

...
jump to index (9800-0): *(this example for a 2d network,99×99)*

8.12.3 Saving the k-mix file

Enter **s** in 8.12 to save the k-mix. A filing prompt box will allow saving a `.mix` file (the default filename is `mymix.mix` (see Filing, chapter 35). The `.mix` can also be saved in section 19.4. The `.mix` can only be loaded in section 8.5. However, information on the k-mix in mixed k networks is contained in the `.w_s`, `.r_s` and `.wrs` files (see section 19.6), so does not need to be saved separately.

8.13 Reviewing the k-mix within "network architecture"

The k-mix may also be reviewed as described in chapter 17 "Reviewing network architecture, wiring and rules". Options in chapter 17 allow changing k, and "neutral" k changes, up to max-k set in section 8.11, which may be greater than the maximum k found in the network. This can be done for single cells, for predefined blocks, or for the whole network.

Chapter 9

The local neighborhood, and network geometry

This chapter defines the CA neighborhood, the pseudo-neighborhood for RBN, the network geometry, and how these are indexed in DDLab.

9.1 The CA neighborhood

In a CA, a cell updates according to the values of it local neighborhood, which usually includes the cell itself and its nearest neighbors. The relative position of the neighborhood cells in relation to the "target" cell is referred to as the "neighborhood template", or just the "neighborhood". For CA the neighborhood is homogeneous throughout the network, and thus requires periodic boundary conditions, where each array edge wraps around to its opposite edge, resulting in a ring of cells in 1d, the surface of a torus in 2d, and a "3-torus" in 3d.

The neighborhood templates used in DDLab, and how they are indexed, are defined below for k=1 to 13, in 1d, 2d and 3d networks.

9.1.1 Triangular and orthogonal neighborhoods - 2d

In 2d, the neighborhoods for k=6 and 7 are based on triangular geometry and result in an effectively triangular network. DDLab displays these as orthogonal networks, which slightly distorts the triangular network structure. For other values of k the neighborhoods and resulting networks are orthogonal.

9.1.2 Symmetry in 2d and 3d neighborhoods

The shapes of 2d and 3d neighborhoods in DDLab are chosen for maximum symmetry. To achieve this, in even k neighborhoods the target cell itself is not included.

9.2 The pseudo-neighborhood, RBN

For RBN, wiring options define how each "target" cell is connected to other cells with respect to its notional or "pseudo-neighborhood", identical to the neighborhood in 1d, 2d and 3d CA. The set of cells that influence a given cell (its actual "neighborhood" to use the term loosely) may be scattered arbitrarily within the network. Each scattered cell is wired to one position in the target cell's pseudo-neighborhood. This allows a CA rule to be applied equally to regular CA and to RBN(see chapter 12).

9.3 1d neighborhood

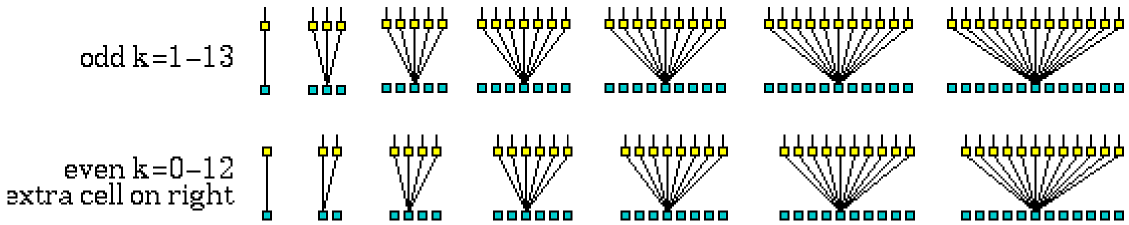


Figure 9.1: 1d neighborhood templates as portrayed in DDLab

the target cell ↓

						0							k=1
						1	0						k=2
					2	**1**	0						k=3
					3	**2**	1	0					k=4
				4	3	**2**	1	0					k=5
				5	4	**3**	2	1	0				k=6
			6	5	4	**3**	2	1	0				k=7
			7	6	5	**4**	3	2	1	0			k=8
		8	7	6	5	**4**	3	2	1	0			k=9
		9	8	7	6	**5**	4	3	2	1	0		k=10
	10	9	8	7	6	**5**	4	3	2	1	0		k=11
	11	10	9	8	7	**6**	5	4	3	2	1	0	k=12
12	11	10	9	8	7	**6**	5	4	3	2	1	0	k=13

Figure 9.2: Neighborhood indexing 1d, k=1 to 13. The "target" cell is shown bold.

The 1d neighborhoods for k=1 to 13 are defined and indexed as shown in figures 9.1-9.2. Cells in the neighborhoods are indexed k-1, k-2,..., 0. For odd k the target cell is centered, for even k the neighborhood is asymmetric with an extra cell on the right.

9.4 2d neighborhood

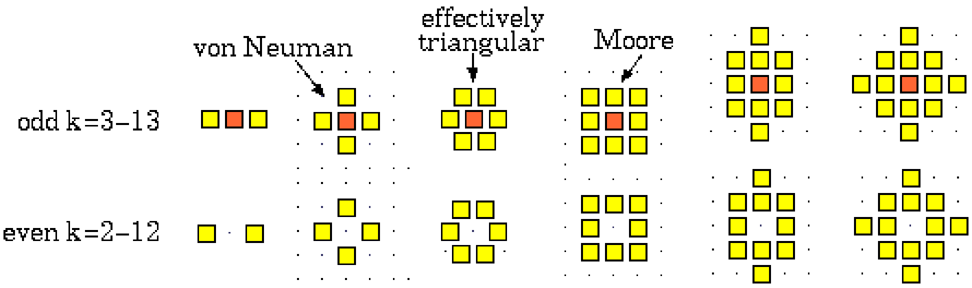


Figure 9.3: 2d neighborhood templates as portrayed in DDLab

k=1

-	-	-
-	**0**	-
-	-	-

k=2

-	-	-
1	$*$	0
-	-	-

k=3

-	-	-
2	**1**	0
-	-	-

k=4

-	3	-
2	$*$	1
-	0	-

k=5

-	4	-
3	**2**	1
-	0	-

k=6 odd

5	4	-
3	$*$	2
1	0	-

k=6 even

-	5	4
3	$*$	2
-	1	0

effect

5	4	
3	$*$	2
1	0	

k=7 odd

6	5	-
4	**3**	2
1	0	-

k=7 even

-	6	5
4	**3**	2
-	1	0

effect

6	5	
4	**3**	2
1	0	

k=6 and *k*=7 neighborhoods produce an effectively triangular grid.

k=8

7	6	5
4	$*$	3
2	1	0

k=9

8	7	6
5	**4**	3
2	1	0

k=10

-	9	-
8	7	6
5	$*$	4
3	2	1
-	0	-

k=11

-	10	-
9	8	7
6	5	4
3	2	1
-	0	-

k=12

-	-	11	-	-
-	10	9	8	-
7	6	$*$	5	4
-	3	2	1	-
-	-	0	-	-

k=13

-	-	12	-	-
-	11	10	9	-
8	7	6	5	4
-	3	2	1	-
-	-	0	-	-

Figure 9.4: Neighborhood indexing 2d, *k*=1 to 13. The target cell is shown bold. *k*=5 is known as the Von Neuman neighborhood, *k*=9 is known as the Moore neighborhood and is used in the "game-of-Life" (see section 16.6). For *k*=6 and 7 the pseudo-neighborhood is different for odd and even rows to define an effectively triangular grid.

The 2d neighborhoods for *k*=1 to 13 are defined and indexed as shown in figure 9.3-9.4. To achieve maximum symmetry, for even *k* the target cell itself is not included in the neighborhood. Cells in the neighborhoods are indexed *k*-1,*k*-2,..., 0. In 2d networks, except for *k*=6 and 7, the neighborhoods define a square grid on the torus, and includes the von Neumann and Moore neighborhoods of 5 and 9 cells. The *k*=6 and 7 the neighborhoods are defined differently for odd and even rows to create an effective triangular grid though it is still represented as orthogonal. For the triangular grid to work, the i and j dimensions of the 2d network must both be even.

9.5 3d neighborhood

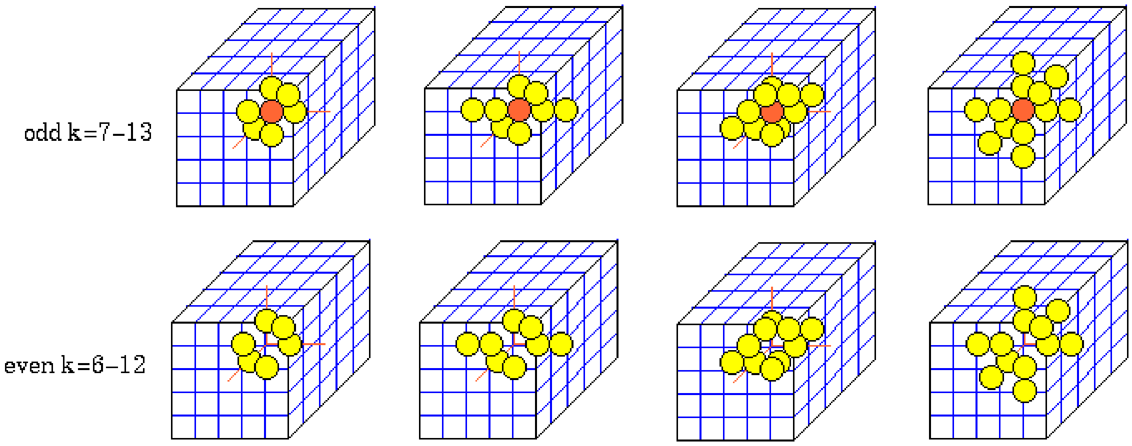


Figure 9.5: 3d neighborhood templates as portrayed in DDLab

```
      5 (up)       6 (up)       7 (up)         8(up)
    4 /          5 /          6 /            7 /
   3 * 2        4 3 2      5 4 * 3 2      6 5 4 3 2
   / 1          / 1          / 1           /  1
 0 (down)      0 (down)     7 (down)       0 (down)
   k=6           k=7          k=8            k=9

                    9 (up)       10 (up)
                   /            /
                8 7 6        9 8 7
                5 * 4        6 5 4
                3 2 1        3 2 1
               /            /
              0 (down)     0 (down)
                k=10        k=11

                   11 (up+)           12 (up+)
                  /                  /
               9  10 (up)        10  11 (up)
               8 /                9 /
           7 6 * 5 4          8 7 6 5 4
             / 3                / 3
    (down) 1  2        (down) 1  2
          /                  /
 (down+) 0   k=12    (down+) 0   k=13
```

Figure 9.6: Neighborhood indexing 2d, k=1 to 13. The k=1 to 5 are the same as for 2d, confined to horizontal i,j layers. Note that the only fully symmetrical 3d neighborhood are k=7 and k=13.

The 3d neighborhoods for k=6 to 13 are defined and indexed as shown in figure 9.5-9.6. To achieve maximum symmetry, for even k the target cell itself is not included in the neighborhood.

In 3d networks, the neighborhoods define a cubic grid on the 3-torus. For k=1-5 the the neighborhoods are the same as for 2d, confined to horizontal i, j layers. For k¿5 the neighborhood penetrates the vertical h axis. Note that the only fully symmetrical 3d neighborhood are k=6 and 7, a 3d nearest neighbor cross, and k=12 and 13, 3 dimensional next-nearest neighbor cross.

9.6 Network geometry

n-1	n-2	n-3	...	2	1	0

1d network, size n

col j-1 ↓						col 0 ↓	
n-1	n-2	n-3	...	...	...	n-i	←row i-1
...	...	...	...	...	...	...	
...	...	...	...	...	...	...	
...	...	...	...	...	...	...	
...	...	...	...	...	...	...	
...	...	...	...	...	...	...	
...	...	...	...	...	...	...	
i-1	...	...	...	2	1	0	←row 0

2d network (size n, $i \times j$)

Figure 9.7: Network indexing, 1d and 2d

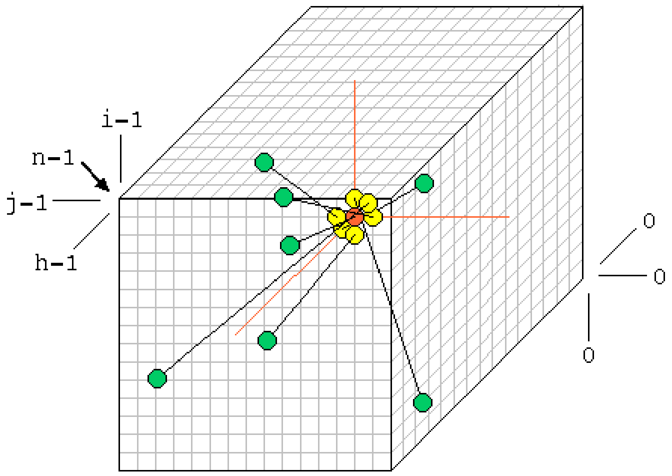


Figure 9.8: Network indexing, 3d, isometric seen from below, showing an example of RBN wiring to the pseudo-neighborhood of a k=7 cell.

Network cells are indexed and arranged in a regular 1d, 2d or 3d space, with axial dimensions i, j, h, as shown in figures 9.7-9.8

This network “geometry” has real meaning for CA, or RBN with various biases, for example where each cell's random inputs are confined close to itself. For RBN with fully random wiring the geometry simply allows convenient indexing and representation.

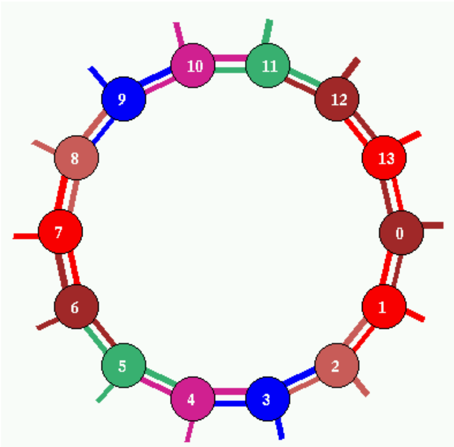


Figure 9.9: A 1d CA, n=14 k=3, a ring of cells, 0-13, shown as a network graph with circle layout (see chapter 20).

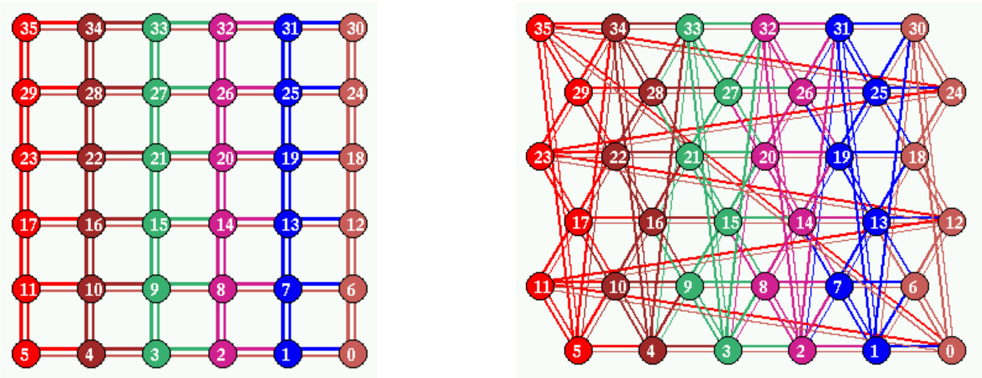


Figure 9.10: 2d CA, n=6x6, cells numbered 0-35. *left*: k=4, with an orthogonal geometry, *right*: k=6, with a triangular geometry. shown as 2d network graphs with (see chapter 20).

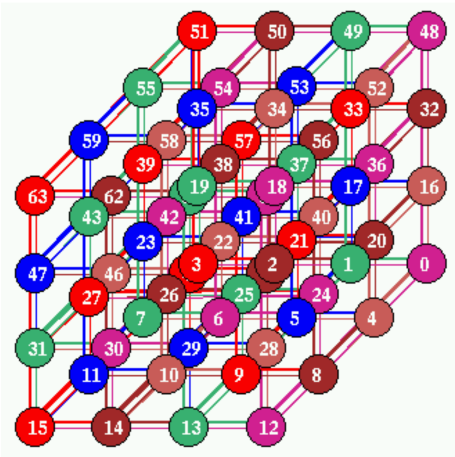


Figure 9.11: A 3d CA, n=4x4x4 k=6, cells numbered 0-63. shown as 3d network graph, (see chapter 20). The graph should be viewed as if looking up from below, were the bottom layer of cells is numbered 0-15).

CA have periodic boundary conditions, where opposite edges join. Local wiring in 1d creates a ring of cells. 2d CA wiring creates a regular orthogonal grid on the torus (except for k=6 and 7, where the grid is effectively hexagonal, though shown as orthogonal). 3d CA wiring creates a 3-torus. "Confined" RBN also have periodic boundary conditions by default, but this can be suppressed.

1d, 2d and 3d networks are indexed as follows,

1d: size n are indexed $n-1\cdots 0$.
2d: size $n = i \times j$ are indexed $i-1, j-1\cdots 0, 0$ and $n-1\cdots 0$.
3d: size $n = i \times j \times h$ are indexed $i-1, j-1, h-1\cdots 0, 0, 0$ and $n-1\cdots 0$..

Figures 9.7 - 9.8 illustrate the geometry of 1d, 2d and 3d networks, and how they are indexed and represented. Figures 9.10 - 9.1.2 show examples of 1d, 2d and 3d CA presented as network graphs (see chapter 20).

Chapter 10

Setting the wiring, quick settings

The main wiring prompts are displayed in context dependent top right windows. The first wiring prompt gives options for quick wiring settings for CA in 1d, 2d or 3d, random wiring for just 1d, or loading a wiring file (see section 10.3. Alternatively, *special* wiring allows more flexible wiring requiring further wiring prompts, described in chapter 11.

10.1 The first wiring prompt

The first wiring prompt is as follows,

> **WIRING: special-s load-l random-r**
> **regular 3d-3 2d-2 1d-def:**

10.2 Special wiring

If **s** is selected in section 10.1 above, prompts are presented for various special wiring options, including setting RBN wiring for 1d, 2d and 3d networks (with various biases), described in chapter 11.

10.3 Loading the wiring scheme

If **l** is selected in section 10.1, filing prompts will allow a `.wir` file to be loaded. Loading such a file resets the network size (see chapter 19).

10.4 Random 1d wiring

If **r** is selected in section 10.1, a wiring scheme will be assigned at random according to the previously selected neighborhood k (and k-mix) settings. The next prompt allows a graphic representation to be displayed, where the wiring to be reviewed and altered (see chapter 17).

To assign random wiring for 2d or 3d networks, or bias the random wiring in various ways, **s** should be selected in section 10.1, and "special" wiring assigned as described in chapter 11.

10.5 Regular 1d wiring

If **return** (the default) is selected in section 10.1, the wiring is set up as regular 1d, with periodic boundary conditions. The network may be a CA, or have mixed neighborhoods and/or rules. The remaining wiring options will be skipped.

10.6 Regular 2d or 3d wiring

If **2** or **3** is selected in section 10.1, the wiring will be set up as regular 2d or 3d. In both cases boundary conditions are periodic, i.e. the 2d array can be imagined as drawn on the surface of a torus, and the 3d array on a 3-torus. The network may be a 2d or 3d CA, or have mixed neighborhoods and/or rules.

As well as setting regular wiring, further prompts will appear to set the 2d or 3d network size and k, or k-mix. Previous size and k settings in the main sequence of prompts will be superseded.

10.6.1 Conserving wiring memory in regular 2d and 3d networks

The following option is presented only if **s** was selected at the very first prompt in chapter 6.1, and restricts the program to showing just space-time patterns, not attractor basins.

forwards only (uses less memory, slower)-y:

If **y** is selected, the 2d or 3d network is not wired up, saving memory that would otherwise be allocated to specify the wiring. The option is useful to run just space-time patters (as in the "game-of-Life") if memory is limited. The "forwards only" run will be slower as each cell's 2d neighborhood must be redefined at each pass. For a system with 8Mb RAM selecting this option should not be necessary.

10.6.2 Setting 2d and 3d network size

The following option sets the $i \times j$ (width $\times$ depth) dimensions of 2d networks, and the $i \times j \times h$ (width $\times$ depth $\times$ height) dimensions of 3d networks.

2d, enter width (def 40): **depth (def 40):**
or
3d, enter width (def 9): **depth(def 9):** **height(def 9):**

Enter the width, depth (and height), keeping in mind the network size limitations (see section 7.4). For space-time patterns the maximum network size, n_{max}=65025, corresponding to a square 2d network, 255×255. The maximum 3d cube is 40×40×40. The network need not be a square or a cubic. The size limits apply so that network positions can be indexed by an unsigned short (16 bits).

If the dimensions selected for a 2d or 3d network exceed 65025, a message such as the following is displayed,

255x256 too big! max size=65025 cont-ret: *(for 2d)*
or
40x40x41 too big! max size=65025 cont-ret: *(for 3d, values shown are examples)*

Pressing **return** restores the network size prompt.

If the intention is to generate attractor basins, the 2d size should be kept very small (i.e. for 2d 3x3 for quick results, 4x4 for slower results).

10.6.3 Set k, or k-mix

The next series of prompts sets k, or the k-mix, as described in chapter 8.

10.7 Networks too big for a basin field

If "basin field" was selected at the first prompt in chapter 6.1, and a network size is set larger than 31, the "basin field" setting will be changed automatically to a single basin or space-time pattern, with the following message,

size too big for basin field (max 31)
changed to single basin or space-time pattern only
continue-ret:

Chapter 11

Setting special wiring

Selecting **s** in section 10.1 allows a greater range of methods for wiring the network. Selecting special wiring is necessary to set random wiring for 2d or 3d networks. The network can be assigned regular or random wiring in 1d, 2d or 3d, or hypercube wiring, and the wiring can be "hand wired" and biased in a variety of ways.

Special wiring allows the following functions, some of which may be restricted to predefined parts of the network (see also chapter 17).

- Confining random wiring within a set zone, from which some wires may be released.
- Suppressing periodic boundary conditions, selectively for the various axes in 2d and 3d.
- Forcing or disallowing self-wiring.
- Forcing distinct wiring i.e. no duplication,
- Setting the same random wiring template for every cell in the network.
- There are additional functions for 3d works.
  - Suppress links to particular layers of the network.
  - Force direct links to specific layers.
  - Apportion fractions of available random links between specified layers.

Once the special wiring is set, it can be ammended in many more flexible ways from the "wiring graphic" as described in chapter 17.

11.1 Setting up the network geometry

The first special wiring option allows the network geometry to be set as 1d, 2d, or 3d, and also as a hypercube for appropriate n and k.

3d-3 2d-2 1d-def:
or
hypercube-h 3d-3 2d-2 1d-def *(if $k = log_2 n$ or $log_2 n + 1$)*

11.2 Hypercube wiring

In a fully connected n-dimensional hypercube, each state receives input from its $log_2 n$ one-Hamming distance "neighbors", and may also receive input from itself. If n=8 and $k = log_2 n = 3$, the hypercube is a simple cube with network states at the vertices and bi-directional links at the edges.

The hypercube wiring option is only active if the network size n is a power of 2, (2, 4, 8, 16...), and $k = log_2 n$, or $k = log_2 n + 1$ where vertices also receive one additional input from themselves. For a simple cube, n=8, k must equal 3 or 4 for the hypercube option to appear.

If **h** for a hypercube is selected in section 11.1 a further option allows degrading the hypercube "wiring", i.e. pruning uni-directional connections, according to some probability,

set % prob (def 100):66 *(for example)*

For example, for an n=8, k=3 hypercube, if 66 is entered, wires are set with a probability of 66%. A further prompt is presented,

part hcube:%wire-prob=66 act=62.5 bi=4/12=33.3
nhood mix=22132321 *(values shown are examples)*
reset-r save-s cont-ret:

This indicates that the degraded hypercube ("part cube") has an actual fraction of remaining wires of 62.5% (this may vary), and that the number of bi-directional links is 4 out of a maximum of 12, i.e. 33.3%. Figure 11.1 gives 3d graph examples.

Enter **r** to reset the wiring probability, **s** to save the k-mix (see chapter 19), and **return** to accept. The resulting k-mix is given for the network as in section 8.12.

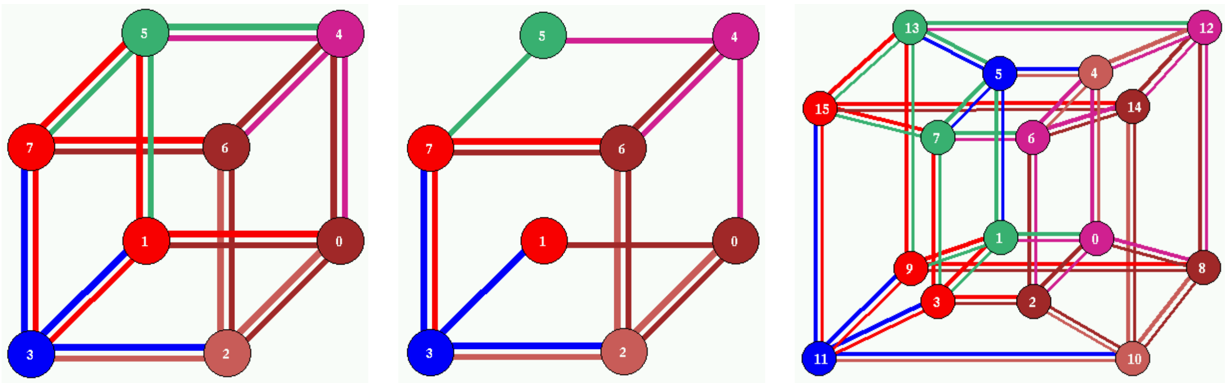


Figure 11.1: *left*: a simple n=8 k=3 hyprecube, and *center*: a degraded version of the hypercube with some links missing. *right*: a n=16 k=4 hyprecube. The 3d network graph option(section 20.4) was used for the figures, for n=16 the graph was rearanged by dragging nodes 20.5). Outputs links are the same color as the source node, and emerge off center clockwise. The figures should be viewed as if looking up from below.

11.3 1d, 2d and 3d special wiring

If **return**, **2** or **3** was entered in section 11.1, a 1d, 2d or 3d network will be selected to receive either regular or random wiring, or "hand wiring" in a wiring matrix "spread sheet".

For regular wiring, the wiring dimension would usually be set to match the network dimension, but its also possible to assign regular 1d wiring to a 2d network, and regular 1d or 2d wiring to a 3d network.

Further network parameters, n, k or the k-mix, and the wiring itself are set subsequently. The following option is presented,

> **hand wire-h,**
> **regular 1d-1, random-def:** *(if 1d was set in 11.1)*
> or
> **regular 2d-2 1d-1, random-def:** *(if 2d was set in 11.1)*
> or
> **regular 3d-3 2d-2 1d-1, random-def:** *(if 3d was set in 11.1)*

Once the network wiring has been set, the wiring of individual cells may be reviewed and altered in a "wiring matrix" or 'wiring graphic" (chapter 17).

11.4 Special wiring, local

Enter **1**, **2** or **3** in section 11.3 for local 1d, 2d or 3d (CA) wiring. Note that for regular 1d wiring in a 1d network, the n, k parameters set previously remain valid (n, section 7, and k, or the k-mix, section 8). For regular 2d and 3d wiring new prompts will be presented to reset these parameters.

11.4.1 Local 1d treated as random

For a 1d network, if **1** for regular 1d wiring was selected in section 11.3, a subsequent option allows the regular wiring to be treated as if it were "random" or non-local (always the case for 2d and 3d networks), allowing the regular wiring to be altered in various ways (and also for wire moves to be allowed in "learning", see chapter 34). However, the "compression" of attractor basins (section 26.1) may be applied as long as the network retains regular 1d wiring. The following prompt is presented,

> **treat regular 1d as random wiring-1, and compress-2:**

11.4.2 Local 2d and 3d

For a 2d or 3d networks, if **2** or **3** for local 2d or 3d (CA) wiring was selected in section 11.3, a further option is presented to conserve wiring memory as described in section 10.6.1. Further prompts are then presented to set the 2d network size, $i \times j$ ($\times\ h$ for 3d), as described in section 10.6.2, and neighborhood k, or the k-mix, as described in section 8. Note that these settings override those previously set for network size n in section 7, and for k in section 8.

Regular 2d and 3d wiring is treated as if it were "random" or non-local, allowing it to be altered in any way. This is not the case by default for regular 1d wiring (see section 11.4.1 above).

11.5 Special wiring, random

If random wiring (the default) is selected in section 11.3, (for 1d, 2d or 3d), and n, k, or the k-mix, have been set, a series of context dependent prompts allow a variety of biases to the random wiring,

bias random wiring: confine to local zone (max=14 def=14): CA-c:
release some wires from zone (def 0, max 3): *(depending on k)*
suppress periodic boundary-s: i: j: h: *(i: j: for 2d, i: j: h: for 3d)*
exclude all selfwiring-2, selfwire centre wire-1:
distinct wiring (no duplication)-n:
suppress links to layers (0-8), range1 low: high: range2 low: high: *(3d only)*
force direct link to layer h, enter (0-8): *(depending on h, 3d only)*
force random links to specific layers-y: *(3d only)*
same wiring everywhere-e:

A small top centre window **accept defaults-d** is a reminder that at any time further prompts in this sequence can be skipped by entering **d**. Enter **q**, or the right mouse button, to backtrack. The options are explained below.

11.5.1 Applying wiring biases to parts of the network

Its important to note that all these biases may be applied to just pre-defined parts of the network as well as for the whole network, which are set from the wiring graphic described in chapter 17

To design particular wiring schemes, especially if the wiring needs to be tailored in specific ways between different parts of the network, its usualy easier to set up a dummy wiring scheme at this stage, them revise it in chapter 17.

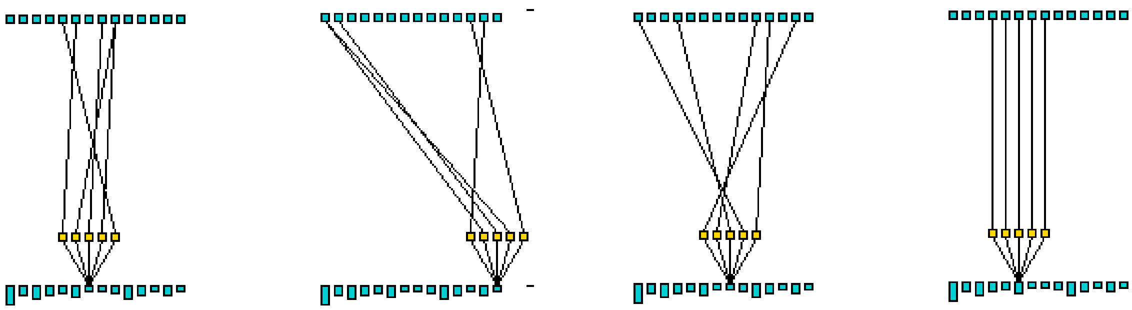
11.5.2 Confining random wiring to a set zone

The wiring may be confined within a periodic zone of a given diameter relative to each target cell. This diameter relates to the network geometry. Enter the local zone diameter at the prompt **confine to local zone:** in section 11.5.

1d, 2d and 3d examples are shown in figures 11.2, 11.3 and 11.3. which illustrate how network wiring is represented, explained more fully in chapter 17.

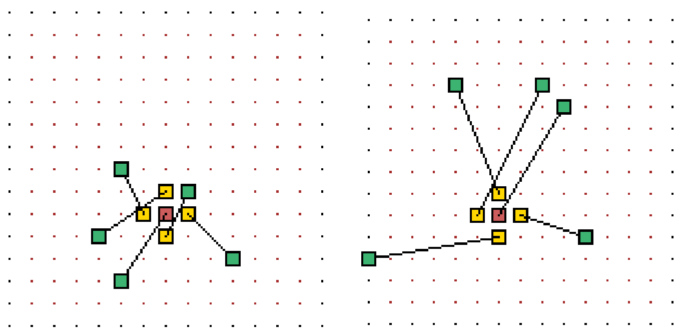
11.5.3 Setting local wiring as random

Enter **c** at the prompt **CA-c:** in section 11.5 to set local, CA, wiring as "random wiring". CA wiring in 1d, 2d or 3d will be set depending on the native dimension of the network. This allows subrequent changes to be made to the CA wiring, for example, releasing some wires described below.



(a) random, zone=5 (b) random, zone=5 (c) random (d) CA wiring

Figure 11.2: Confining 1d random wiring within a local zone of a set diameter. (a) and (b) show random wiring within a 5 cell zone, where (b) illustrates periodic boundary conditions. (c) shows fully random wiring, and (d) local, CA type, wiring for comparison. Wiring from the pseudo-neighborhood is shown connected to to the previous time-step (see chapter 17)



(a) random, zone=7 (b) fully random

Figure 11.3: Confining 2d random wiring within a local zone of a set diameter. (a) random wiring confined within a 7 cell zone. (c) fully random wiring. Wiring is shown to the pseudo-neighborhood

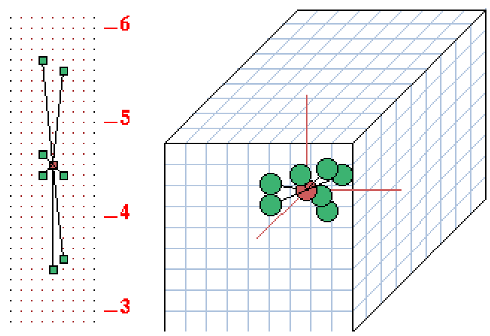


Figure 11.4: Confining 3d random wiring within a local zone of a set diameter. Represented in 2d showing successive layers(left), and in 3d (right). The pseudo-neighborhood is not shown.

11.5.4 Release wires from zone

Some wire can be released from the local zone set in section 11.5.2 and from the CA wiring set in section 11.5.3. Enter the number of wires to be released at the prompt **release some wires from zone (def 0, max 7):** *(for example)* in section 11.5. The number of wires specified will be chosen and reassigned at random to any position in the network, unless this is restricted by further biases.

11.5.5 Suppress periodic boundary conditions

For a 1d network, enter **s** at the prompt **suppress periodic boundary-s:** in section 11.5 to suppress periodic boundary conditions. For 2d and 3d networks further prompts appear, **i:**, **j:** (and **h:** for 3d). Periodic boundary conditions can be suppressed independently for each axis, i, j and h. To do so, enter **s** in responce to the prompts.

11.5.6 Self-wiring

Enter **2** or **1** at the prompt, **exclude all selfwiring-2, selfwire centre wire-1:** in section 11.5 to exclude self-wiring, or to force target cells to wire to themselves. Self-wiring is made from the central pseudo-neighborhood position for odd k, and the near central position (the higher neighborhood index) for even k (chapter 9 describes how the neighborhood is indexed).

11.5.7 Distinct wiring

Wiring may be made distinct to prevent two or more positions in the pseudo-neighborhood to be wired to the same cell. Enter **n** at the prompt, **distinct wiring (no duplication)-n:** in section 11.5 for distinct wiring. DDLab will do its best to achieve distinct wiring, but this may conflict with a small "wiring zone" set in section 11.5.2 and with other wiring bias options.

11.5.8 Suppress links to 3d layers

Links to horizontal layers in a 3d network can be suppressed. Two ranges of layers can be designated. Prompts are presented in sequence. Enter the levels to be suppressed starting with the lowest level, for example,

suppress links to layers (0-8), range1 low:0 high:3 range2 low:6 high:6

would suppress links to layers 0,1,2,3 and 6.

11.5.9 Force a direct link to a 3d layer

Direct links can be "forced" to each cell in a designated 3d layer. A direct link means that cells at position i, j link one wire (from the pseudo-neighborhood index 0) to position i, j in the designated layer. The wire originates from the pseudo-neighborhood index 0 (see chapter 9).

For example, for a 9 layer 3d network, enter the required layer index at the prompt, **force direct link to layer h, enter (0-8):** to which direct links will be forced. By default, if a layer is specified, all cells make a direct link, but this can be restricted to a specific cell, range of cells, or to a specific layer or range of layers from the wiring graphic (see chapter 17).

This option may be used to provide a constant input pattern to a 3d network, for example to model inputs from a "retina".

11.5.10 Force random links to specific 3d layers

Fractions of the available random links can be assigned to designated 3d layers. Previous biases to confine wiring to a set zone will be respected for the horisontal plane, but the vertical constraints may be overridden.

If **y** is entered at the prompt **force random links to specific layers-y:** in section 11.5, the following series of options are presented in a top right window, for example in an 9 layer, k=7, 3d network,

k=7, enter n links to layer 8 (7 available):3 *(3 entered, 4 remain)*
k=7, enter n links to layer 7 (4 available):1 *(1 entered, 3 remain)*

...
k=7, enter n links to layer 0 (3 available):3 *(none remain)*

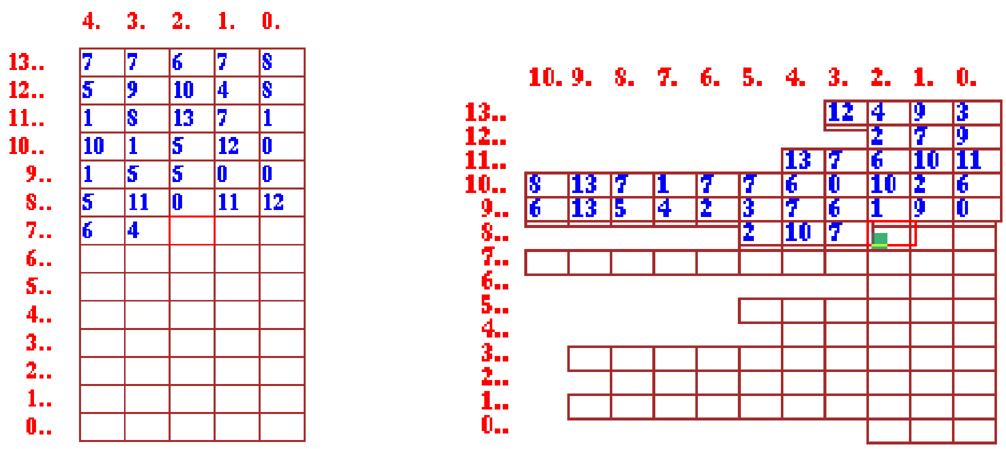
The prompts continue until layer 0, or until all links have been assigned and none remain. To miss a layer enter **return** or **0**. If links remain at the end, they will be assigned at random, but still according to the biases specified in section 11.5.

11.5.11 Same random wiring everywhere

The same random wiring "template" can be applied to every cell in the network to create a quasi-CA with periodic boundary conditions. To do this enter **e** at the prompt **same wiring everywhere-e:** in section 11.5.

Biases previously specified in section 11.5 and other options will be respected as much as possible.

11.6 Wiring by hand



(a) k=5, n=14 (b) k= 3 to 11, n=14

Figure 11.5: Setting wiring by hand on the blank wiring matrix, shown partly filled. k-1...0, indexes columns, n-1...0, indexes rows. (a) shows an example matrix for a k=5, n=14 network. (b) shows an example matrix for a mixed k network, k=3 to 10, n=14.

If **h** is selected in section 11.3, a blank wiring scheme is presented in the form of a matrix or "spread sheet", which may be filled in with the wiring positions for each cell's pseudo-neighborhood index. A completed wiring matrix can also be amended as described in section 17.2.

Columns give the cell's pseudo-neighborhood index, K (k-1...0), Rows give the cell network position, N (n-1...0). The 0-0 grid, or the 0-minimum n grid if the whole matrix does not fit within one window, is in the lower right hand corner. Each grid records the position in the network (n-1...0), to which the K'th wire of the N'th cell is connected.

Move around the matrix with the left/right/up/down arrow keys. Enter the new position at the flashing green cursor and complete the entry by moving to another grid with an arrow key or entering **return**. On a blank grid, or on 0, **return** gives a random position. An entry outside the network limits will be ignored. Enter **q** to complete. Any undefined grids will be set to position 0. While the wiring matrix is being set, the following prompt and reminder is displayed in a top right window,

> **hand wire/revise: jump-j** *(values shown are examples)*
> **enter wiring positions 0-144 (return on blank/0=random)**
> **move-arrows more/complete-m layout-l font-f quit-q**

Enter **l** or **f** to alter the presentation of data and the amount visible in the matrix window. **f** toggles the font size between normal and small. **l** changes the matrix window width. The following top right prompt is presented,

> **change window width (922-231 now=462):** *(values shown are xamples)*

Enter the new width in pixels within the limits indicated.

Large networks may require several successive windows to display the matrix. Enter **m** to see the next window. Enter **j** to jump to a new cell index, the following prompt is presented,

> **jump to index (1599-0):** *(for a 2d network, 40×40)*

Enter the new cell index, which will become the first entry in the top row. Moving outside the first or last entry in the current window with the arrow keys or return also brings up new window.

Enter **q** to accept the wiring and go to the prompt in section 11.7 below.

The wiring scheme can be reviewed and revised in the same wiring matrix format, or as a wiring graphic in 1d, 2d or 3d, as described in chapter 17.

11.7 Reviewing wiring

After the special wiring has been set as described in this chapter, various options allow the wiring to be reset, reviewed and ammended from a wiring graphic or wiring matrix.
A prompt similar to the following is presented,

> *(for a 1d network)*
> **1d network (n=150),review/revise, wiring only - rules not set**
> **graph-g, matrix: revise-m view=M**
> **graphic: 1d:timesteps-1 circle-c, 2d-2:**
> *(for a 2d network)*
> **2d network (40x40), review/revise, wiring only - rules not set**
> **graph-g, matrix: revise-m view=M**
> **graphic: 1d:timesteps-1 circle-c, 2d-2:**
> *(for a 3d network)*
> **3d network (15x15x15), review/revise, wiring only - rules not set**
> **graph-g, matrix: revise-m view=M**
> **graphic: 1d:timesteps-1 circle-c, 2d+3d-3:**

These options are described in chapter 17. They provide very flexible methods to review and ammend rules (once set) as well as wiring, and it may be preferable to set up a suitable dummy network initialy, then tailor it with these options.

If just *quick* wiring was set in chapter 10, these options do not appear at this point in the program, but they appear in any case after the rules have been set (chapters 12-16).

Chapter 12

Rules

This chapter describes the system of logic, or rules, that act on the pseudo-neighborhood to determine a cell's output. The rules are Boolean functions. In DDLab they are expressed in the most general form, as a look-up table or rule-table. The same rule may apply to all cells in the network as for CA, or the network may have a mix of different rules which may be assigned and modified in various ways.

12.1 The rule-table

A CA or RBN rule is defined by a lookup-table or rule-table. A neighborhood of size k has 2^k alternative (0,1) configurations. Table 12.1 shows how the size of the rule-table increases exponentially with k.

The convention in DDLab is to list neighborhood configurations from left to right in reverse value order, with the all 1's neighborhood on the left. Each neighborhood is assigned an output (0,1), and the resulting bitstring defines one of the 2^{2^k} possible rules. The most significant bit in the bitstring corresponds to the all 1's neighborhood. This convention can be inverted (see section 18.3). The rules may also be expressed in decimal (for $k \leq 5$) or in hexadecimal. $k \geq 5$ rules are refered to by their hex rule numbers, $k \leq 3$ rules are usualy refered by their more familiar decimal rule numbers.

Setting out the neighborhood configurations verticaly gives a rule-table "neighborhood matrix". For example, the k=3 neighborhoods are set out verticaly as shown below. The output bitstring defines the rule.

```
                 7......0 - rule-table index
                 :      :
             2 - 11110000
     k index 1 - 11001100
             0 - 10101010
                 --------
                 11000001 - output
                            = rule 193 in decimal, c1 in hex
```

The k=5 neighborhoods are set out verticaly as shown below. The output bitstring defining an example rule.

```
             31...... ........ ........ .......0 - rule-table index
             :                                 :
         4 - 11111111 11111111 00000000 00000000
         3 - 11111111 00000000 11111111 00000000
 k index 2 - 11110000 11110000 11110000 11110000
         1 - 11001100 11001100 11001100 11001100
         0 - 10101010 10101010 10101010 10101010
             -------- -------- -------- --------
             11111100 01100010 10001000 00110111 - output
                                                   = rule fc 62 88 37 in hex
                                                   = rule 4234315831 in dec
```

In DDLab a graphic of the rule-table matrix (figure 12.1 is displayed as a reminder (for k=1 to 9 only, and not for mixed k).

12.2 The totalistic code-table

"Totalistic" rules are rules where a cell's value depends only on the sum of 1s in its neighborhood. The totalistic "code" is defined by a lookup table of k+1 possible totals set out in reverse value order from k to 0. Each total is assigned an output (0,1). The resulting bit string defines each of the 2^{k+1} possible totalistic codes. DDLab automatically transforms the code into the rule-table format.

For example, the k=5 totals are set out thus,

```
5 4 3 2 1 0 - totals
- - - - - -
1 1 0 0 1 0 - output
              = code 50 in dec, 32 in hex,
              = rule 3900801302 in dec, e8 81 81 16 in hex
```

Totalistic codes are also useful for setting threshold functions, for example, the totalistic code `111000` gives the k=5 majority rule.

12.3 Totalistic rules

After the network wiring has been set or defaults accepted (in chapters 10 - 11), the next prompt (in the main sequence) allows just totalistic rules to be selected,

totalistic rule-t:

If **t** is entered, subsequent rule settings (including random settings) will be confined to the totalistic rules only. DDLab automatically transforms the totalistic code into the rule-table format.

12.4 Rule-table matrix

A graphic representation of the rule-table neighborhood matrix is displayed as a reminder, for k=1 to 9 only (and not for mixed k), as shown in figure 12.1, where the size of the rule-table equals 2^k:

k	1	2	3	4	5	6	7	8	9	10	11	12	13
size of lookup table 2^k	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192

Table 12.1: The size of rule-tables k=1 to 13

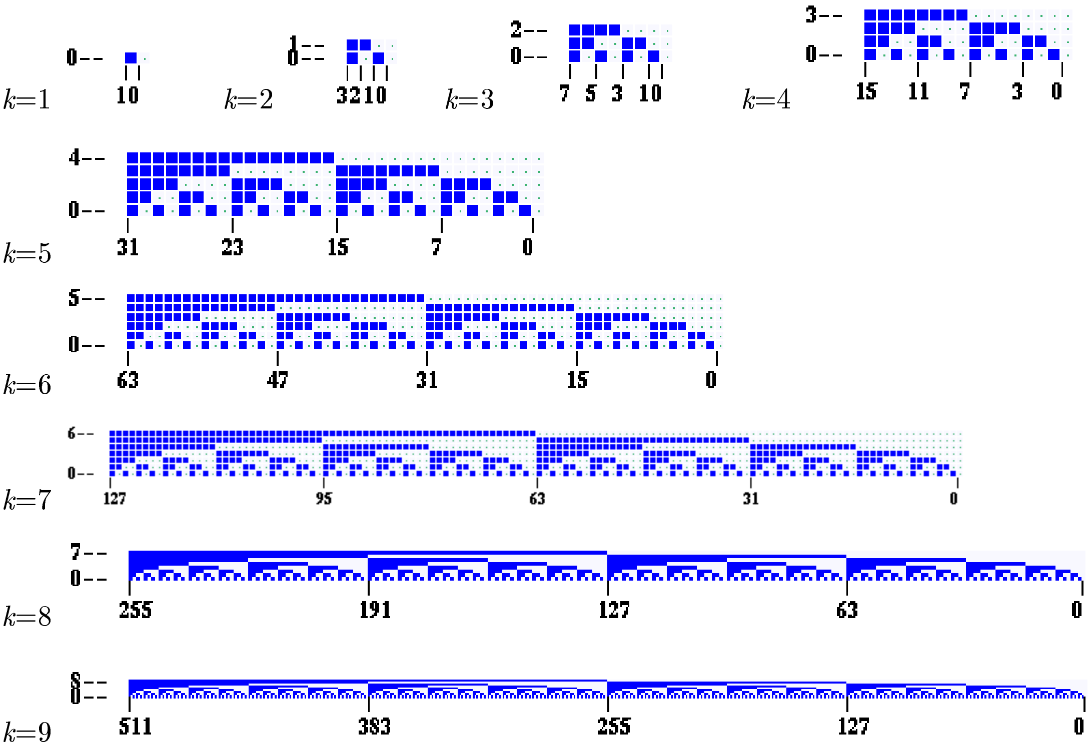


Figure 12.1: The rule-table neighborhood matrix k=1 to 9, as displayed in DDLab

12.5 Single rule or rule-mix

All cells in a network may have the same rule (as in CA), or cells may have different rules, a heterogeneous rule-mix. A rule-mix can subsequently be set at random from the whole of rule-space (i.e. with *no limit*), or provided there is no neighborhood mix (k-mix), a limited subset of rules can be specified from which the random allocation of rules is made. The rule-mix (or rule-scheme) may be loaded from a `.r_s` file (see chapter 19).

Provided there is no k-mix, the following top right prompt is presented,

RULES: single rule (def), load rulemix-l *(or* **single code...**
mix: no limit-n, or set limit up to 200: *(...if totalistic was selected in 12.3)*

Enter the following,

- **return** for a single rule (or code). See chapter 16 for further options.
- **l** to load a `.r_s` rule-mix (rule-scheme) file See chapter 19.
- **n** for a rule-mix (or code-mix) taken from the whole of rule-space. See chapter 13 for further options.
- a number (1-200) for setting the size of the limited subset of rules (or codes) from which the rule-mix (or code-mix) will assigned to the network. See chapter 13 and 14 for further options.

12.6 A rule-mix with just one rule

To set up regular CA architecture that allows later manipulations as if it had a rule-mix, select a rule-mix with a limited subset of rules consisting of just one rule (i.e. enter **1** at the prompt in section 12.5, and set the rule subset *by hand* (section 13.6) This fools DDLab into assuming the network has a "rule-mix" where in fact all rules are the same.

Chapter 13

Rule-mix options

This chapter describes methods for setting a heterogeneous mix of rules in the network. A rule-mix is selected if **n**, or a number (1-200), is entered at the prompt in section 12.5.

n gives a rule-mix (or code-mix) taken from the whole of rule-space.

A number (1-200) sets the size of a limited subset of rules (or codes) from which the rule-mix (or code-mix) will be assigned to the network, described further in chapter 14.

A series of further top right prompts are presented to set the actual rule-mix (or code-mix), which may be biased in various ways.

13.1 The all 0s output

Rules set at random may be biased to have the "all 0s" neighborhood output set to 0 (the default), or any rule may be allowed, The following prompt is presented,

if random, allow any rule-a, or force all 0s->0 (def):

Enter **a** to allow any rule, or **return** to accept the default.

In a locally wired network, forcing all 0s->0 ensures that a small zone of 1s can grow only gradually, otherwise 1s can appear at the first time-step throughout the network.

13.2 Setting the neighborhood, k=0

Although it is illegal in DDLab for a cell to have zero input wires, k=0, this can nevertheless be easily achieved by setting k=1 with the cell wired to itself, and setting the rule to (dec) 2, or 10 in the rule-table. This results in an effective wiring of k=0 (see section 8.2).

If rules are to be set at random, DDLab identifies any self-wired k=1 cells, and gives an option to set the rule at these cells to make them effectivly k=0. The following prompt is presented,

single self wirings=4, set rule for k=0 equiv-0: *(the number of k=1 self-wired rules)*

Enter **0** accept.

Note that if the effective wiring to a cell is k=0, it will appear disconnected in the 1d wiring graphic (see section 17.4), as illusrtated in figure 13.1.

Figure 13.1: Effective neighborhood k=0, as shown in the 1-d wiring graphic (see section 17.4)

13.3 Methods for setting the rule-mix

If **n** for *no limit* was selected in section 12.5, options allow the rules to be set entirely at random, at random from the "Altenberg" rules (see section 16.5) or the "chain" rules (see section 16.7), or with various other constraints. Rules may be set individually by the various methods described in chapter 16 Rules may be set at random, then biased to include a fraction of "canalizing" inputs. The following prompt is presented,

select by hand-h, Alt-A chain-c rand (def)
random + canalizing-c:

For a mixed k network (see chapter 8), an additional option is offered to set "majority" rules (optionally with their end bits flipped),

select by hand-h, maj-m, +flipped-f, Alt-A chain-c rand (def)
random + canalizing: C or +C:

These option are described below.

13.4 Rule-mix - random

The following options in section 13 assign rules at random,

option	...	*what it means*
rand (def)	...	Enter **return** to assign rules entirely at random from the whole of rule-space.
Alt-A	...	Enter **A** to assign rules at random from the subset of "Altenberg" rules (see section 16.5).
chain	...	Enter **c** to assign rules at random from the subset of "chain" rules (see section 16.7).

The program then continues with options to review network architecture described in chapter 17.

13.5 Rule-mix - random + canalizing

If **C** is entered in section 13, rules will be assigned at random, and the program will continue with options to view and revise the "canalising" inputs in the rule-mix, described in chapter 15.

The program will also continue with the canalising options if another entry is made followed by **C**, for example **AC** for "Altenberg" rules or **cC** chain rules.

13.6 Rule-mix by hand

If **h** is selected in section 13.3 above, the method of rule selection is first set, then the rule at each cell index will be set in turn, starting from cell 0, as described in chapter 16. A reminder keeps track of the cell index, for example,

index-6 (nhood=3 rulemix), default selection is d

During the procedure its possible to jump to another cell index or change the selection method. See section 14.4), where the same methods apply and are described more fully.

13.6.1 Repeating rules (automatically) for mixed k

While setting rules by hand for a mixed k network, an optinon allows a rule that was set for the last k-cell to be repeated for the current k cell, or to be repeated automatically for all the remaining k-cells in the network. The following top right prompt is presented, for example for a k=5 cell at index 4, where k=5 has occured previously,

index-4 (nhood=5 rulemix), default selection is r
set this k=5 cell same as last-s, all remaining-a:

Enter **s** to copy the rule from the last occurring k-cell.
Enter **a** to repeat rule from the last occurring k-cell to the current and remaining k-cells.

13.6.2 Mixed k where all k's (and rules) are the same

To create a mixed-k network where all k's *and rules* are the same, at the prompt for setting the k-mix in section 8.4,

set nhood MIX: load-l by hand-h specify-s random (def):

select **s**.
Then set the percentage of the k required to 100% in in section 8.8.1, for example,

enter % this nhood size (k=3 100.0% left) back-b:100

To make it possible in the future to increase k for some cells, set max k at some higher value in section 8.11, for example,

... **set max k (def 3):7** *(to make max k=7, though this does not occur in the network)*

All the rules in the network may now be made the same by selecting **h** in 13.6, then repeating the first rule automaticaly for the whole network as in 13.6.1.

13.7 Rule-mix - majority

For a mixed k network, if **m** is selected, in section 13.3, the "majority" rule will be set for each cell according the its neighborhood size. The majority rule outputs 1 for a majority of 1s in the neighborhood, 0 for a majority of 0s. In the case of even k, and a tie between 1s and 0s, the output is set according to the value of the central neighborhood index, as defined in chapter 9.

13.8 Rule-mix - majority + end bits flipped

For a mixed k network, if **f** is selected in section 13.3, "majority" rules are set as in section 13.7 above, but the "end bits" in the rule table (i.e. the entries for the neighborhoods all-1s and all-0s) are flipped (complimented). This can lead to some interesting "bi-stable" dynamics.

13.9 Rule-mix for large networks, or large k

Assigning the rule-mix to large networks, especially for large k, may take some time. For $n \geq 5000$ or ($n \geq 500$ and $k \geq 10$), the assignment is monitored by a progress bar near the top right hand corner of the screen.

When the assignment is complete, the program will continue with options to review network architecture described in chapter 17.

Chapter 14

Specify a subset of rules

This chapter does not apply to mixed k networks.

A limited subset of rules (or codes) may be specified from which the rule-mix (or code-mix) will be assigned to the network. The procedure is started by selecting the number of rules in the subset (maximum 200) in section 12.5.

14.1 Set the rule-set by hand or at random

If a number (1-200) was entered at the prompt in section 12.5, then (following various other prompts in section 13.1 and possibly 13.2) an option is presented to set the rules in the rule-set individually (*by hand* by the various methods described in section 16), or to set the rules in the rule-set at random,

> **limited ruleset (12), set by hand-h, at random (def):**
> *(depending on the "limit" set in section 12.5)*
> or
> **limited codeset (12), set by hand-h, at random (def):**
> *(if totalistic codes were selected in 12.3)*

14.2 Setting the rule-set at random

Enter **return**, the default, in section 14.1 above, to assign the rule-set at random. The program will continue with options to review network architecture described in chapter 17.

14.3 Setting a rule-mix or rule-set by hand

Note that this proceedure is essentially the same as setting a "Rule-mix by hand" for each cell index, as described in section 13.6.

If **h** is selected in section 14.1, the method of rule selection is first set, then the rule at each rule-set index (or cell index) will be set in turn, starting from index 0, as described in chapter 16.

The following reminder is displayed,

select subset of 12 rules *(depending on the "limit" set in 12.5)*
or
select subset of 12 totalistic codes *(if totalistic codes were selected in 12.3)*

The various methods of "by hand" selection are described in Chapter 16, *Setting a singe rule.* The rule at each rule-set index (or cell index) is set in turn, starting from index 0. A reminder keeps track of the index, for example,

index-6 (k=5 rulemix), default selection is r *(values shown are examples)*

The prompt is repeated for each successive rule-set index (or cell index), and each rule is selected by the methods described in chapter 16. During the procedure its possible to jump to another cell index or change the selection method, see 14.4 below.

Once the rules in the rule-set have been set by hand, they are assigned at random to the network. The program will continue with options to review network architecture described in chapter 17.

14.4 Changing the cell index or selection method

If **q** is entered at the *select rule* prompt in chapter 16, a top right window is presented with a prompt that allows a jump to a new cell index, to restart, or to change the default rule selection method,

change: index8-i, start again-a, selection-s, cont-ret: *(values shown are examples)*

14.4.1 Changing the cell index

If **i** is entered in section 14.4 above, the following prompt allows a jump to a new cell index.

this index=3, enter new index 0-15: *(values shown are examples)*

If **a** is entered in section 14.4, the cell index is reset to 0.

14.4.2 Changing the rule selection method

If **s** is entered in section 14.4, the following prompt allows the rule selection method to be changed (the precise wording is context dependent and may vary considerably).

index=5 (k=9), default selection=r, to change enter *(values shown are examples)*
maj-m life-L random-r bits-b hex-h repeat-p load-l:

Enter the required selection method, which becomes the the default.

14.5 Assigning the rule-set to large networks, or large k

Once the rules in the rule-set have been set, either hy hand or at random, they are assigned at random to the network.

As in section 13.9, assigning the rule-set to large networks, especially for large k, may take some time. For $n \geq 5000$ or ($n \geq 500$ and $k \geq 10$), the assignment is monitored by a progress bar near the top right hand corner of the screen.

When the assignment is complete, the program will continue with options to review network architecture described in chapter 17.

Chapter 15

Setting Canalization in a random rule-mix

This chapter describes methods for biasing a random rule mix by including varying proportions of "canalizing" inputs. This biases a network towards order, with applications in modeling genetic regulatory networks[4, 17].

A canalizing input may be defined as follows: If a particular value (0 or 1) on an input to a cell (referred to as a "gene" in this context) determines the gene's output irrespective of its other inputs, that input is said to be canalizing. A given gene may have from zero to k canalizing inputs[1]. As k increases, the fraction of rule-space with canalizing rules (having at least one canalizing input) deceases exponentially (see 24.6), so the probability of setting such a rule at random deceases accordingly.

The methods bias the randomly assigned rules by varying or tuning the fraction of canalizing rules or inputs in the network. The algorithms for tuning canalization in DDLab are designed to minimize secondary biases in the distribution of canalizing inputs. Rules tables will be amended at random for the required degree of canalization.

15.1 Selecting Canalizing

Canalizing can be selected at two alternative places in DDLab.

When setting a rule-mix in section 13.3, at the prompt,

> **select by hand-h, random (def)**
> **random + canalizing-c:**

Enter **c** to select canalizing.

Alternatively, once the rule-mix has been set, canalizing can be reached from many points in DDLab from the wiring graphic prompts described in chapter 17. At the prompt,

> **...**
> **... rule: rev/trans-v/t**
> **...**

[1] The outputs of all canalizing inputs to a given gene must be the same

Enter **t** to get the *transform rule* options described in chapter 18. A top right option window similar to the one below will be presented,

> **transform rule: solid-o invert-v comp-c neg-n ref-r canal-C (all+a)**
> **equiv>k (4-7), max k-m, eff k: all-K this-k, save-s:**

Enter **Ca** to select canalizing for the whole network, described in this chapter, or just **C** to set canalizing for a single rule described in section 18.7.

15.2 The first canalizing prompt

A series of top right prompts allow the canalization to be set and reviewed. The first canalizing prompt is as follows,

> **set canalizing genes-g inputs-(def):**

Enter **g** to specify canalizing genes, i.e. the fraction of network elements, n, with at least one canalizing input.

Enter **return** (the default) to specify canalizing inputs (wires), where the total number of wires is $n \times k$ for homogeneous k, or the sum of all k's for a mixed k network.

15.3 Canalizing percentage or number

For a network with homogeneous k, according to whether "genes" or "inputs" were selected in section 15.2 above, the next prompt is as follows,

> **c-inputs: redo-q number-n %-(def):** *(genes/inputs depends on the earlier choice)*
> or
> **mixed *k*, c-inputs: redo-q number-n %-(def):** *(for mixed k networks)*

Enter **return** (the default) to set a percentage of the total genes/inputs as canalizing, or enter **n** for a specific number. From this point on, the options for networks with homogeneous k, and mixed k, are somewhat different.

15.4 Canalizing - homogeneous k

The next prompt gives the fraction and percentage of the current canalizing genes and inputs, and asks for the required canalizing settings, for example,

> **c-inputs=0/4500=0.0% genes=0/900=0.0%** *(for a 2d network, k=5, 30×30)*
> **set new % c-inputs, redo-q accept-ret:**
> *(number/% and genes/inputs depends on the earlier choices)*

If, say, **45** (for 45%) is entered. The prompt will reapper with updated information,

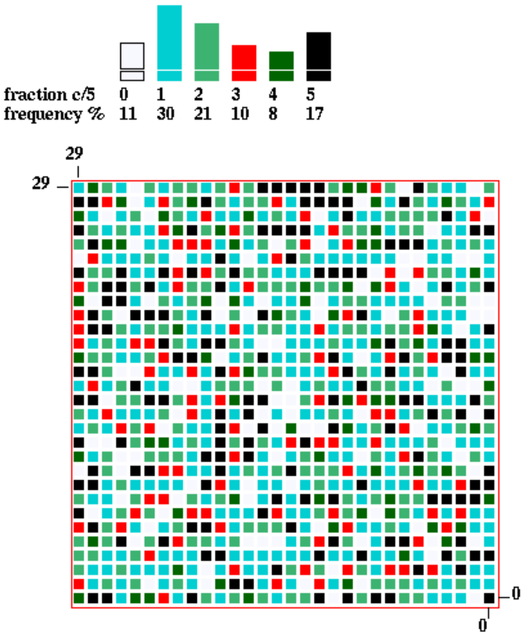
c-inputs=2025/4500=45.0% genes=793/900=88.1%
set new % canalizing inputs, redo-q accept-ret:

You may enter a new canalizing setting, or enter **return** to accept, the program will continue with options to review network architecture described in chapter 17. Enter **q** to revert to the first canalizing prompt (15.2).

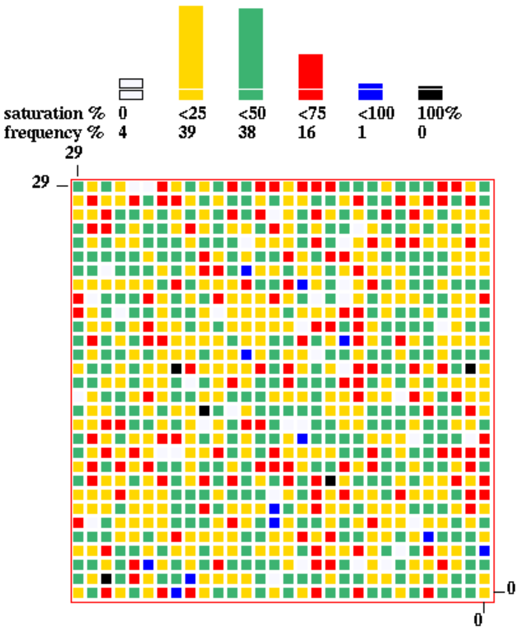
At the same time as the prompt appears, a window in the lower right hand corner of the screen gives a graphical display of network canalization (figure 15.1). This shows a 2d representation of the network (irrespective of its native dimension) showing the degree of canalisation of each gene.

Above this is a histogram of canalizing frequency for $k \leq 9$, or canalizing saturation for $k \geq 10$ (also for mixed k, see section 15.5).

The frequency histogram shows each degree of canalization, 0-k (see figure 15.1). The saturation histogram shows the frequency of different percentages of canalization, for 0%, 100%, and intervals of 25% in between, i.e. $0, < 25, < 50, < 75, < 100, 100$ (see figure 15.2).



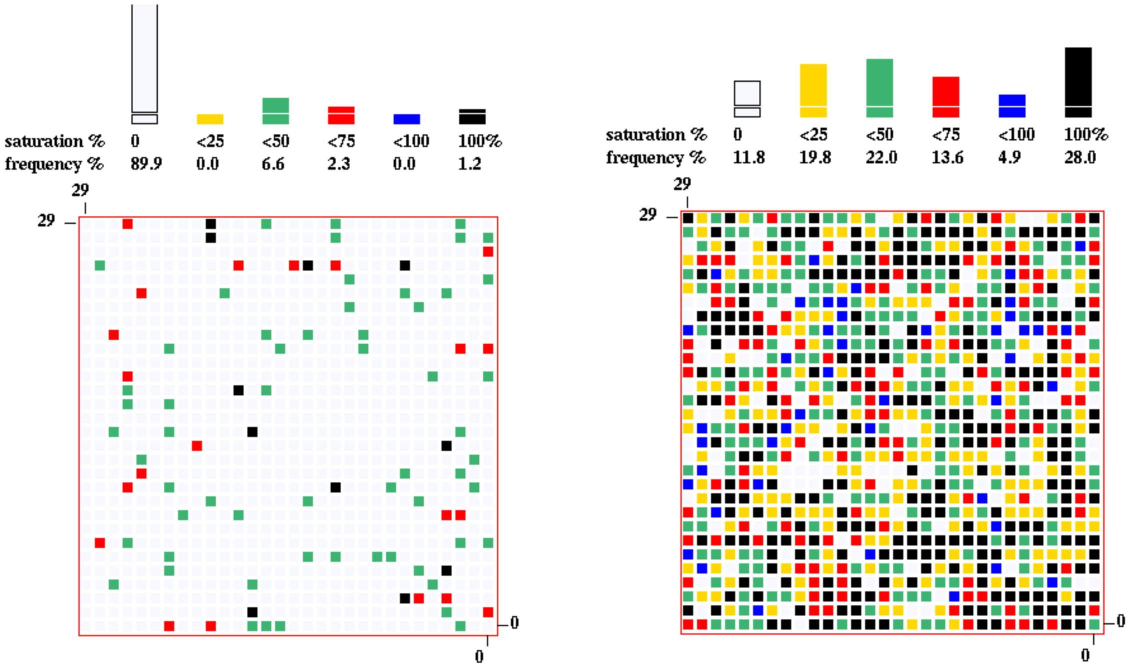
k=5, canalizing fraction, canalizing inputs set to 45%.



k=10, canalizing saturation, canalizing inputs set to 20%.

Figure 15.1: Canalizing frequency/saturation for homogenious k networks, n=30×30. *left*: The fraction of canalizing inputs of each gene in the network, k=5, and a histogram of the frequency of each fraction. The canalizing inputs set to 45%. *right*: The saturation of canalizing inputs of each gene in the network, k=10, and a histogram of the frequency of each saturation. The canalizing inputs set to 20%. The colors in the network and histogram correspond.

15.5 Canalizing - mixed k



Initial canalizing inputs, 3.0%. Canalizing inputs set to 45%.

Figure 15.2: Canalizing saturation for two networks with the same (roughly evenl) k-mix, 3-7, as shown in section 15.5. n=30×30. *left*: The initial network showing initial, expected, canalizing. *left*: The canalizing inputs set to 45%. The colors in the network and histogram correspond.

For a network with mixed k, following the prompts in sections 15.2 and 15.3, the next prompt shows the percentage of different k's (see also section 8.8.1), and offers an option to reset the canalization for just a subset of the network having a particular k, or for the whole network,

> ...
> **%k: 3=19.9 4=19.7 5=19.1 6=21.0 7=20.3**
> **enter k for just one nhood, all-(def)** *(genes/inputs depends on the earlier choice)*

15.5.1 Canalizing for the whole network - mixed k

If **return** is entered at the prompt in section 15.5 above, the next prompt gives the fraction and percentage of the current canalizing genes and inputs, and asks for the required canalizing settings.

For example, the prompt may appear as follows, in this case for a 30×30 network, with a k-mix of 3 to 7,

> **c-inputs=134/4530=3.0% genes=91/900=10.1%** *(2d network, 30 × 30)*
> **set new % canalizing inputs, redo-q accept-ret:**
> *(number/% and genes/inputs depends on the earlier choices)*

The reason for the initial canalising degree is that about 23.5% of inputs in randomly assigned k=3 rules are likely to be canalising. For k=4 rules the figure is about 1.5(section 24.6 describes how these rule-space properties are derived).

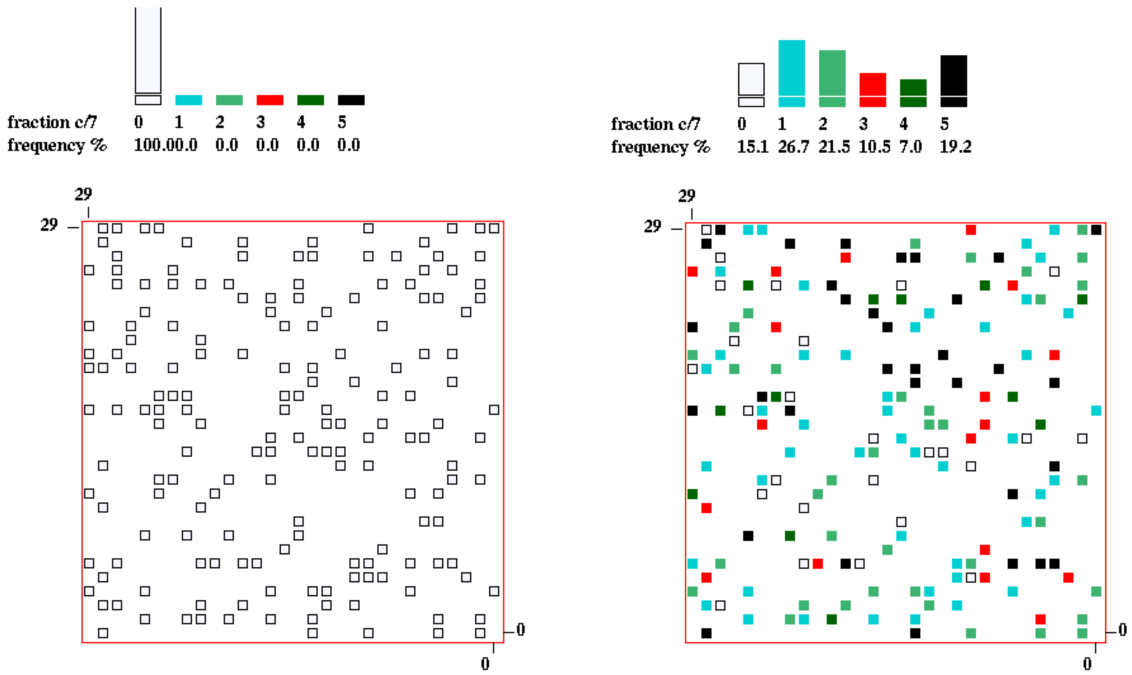
If, say, **45** (for 45%) is entered. The prompt will reapper with updated information,

> **c-inputs=2034/4530=45.0% genes=794/900=88.2%** *(2d network, 30 × 30)*
> **set new % canalizing inputs, redo-q accept-ret:**
> *(number/% and genes/inputs depends on the earlier choices)*

You may enter a new canalizing setting, or enter **return** to accept, the program will continue with options to review network architecture described in chapter 17. Enter **q** to revert to the first canalizing prompt (15.2).

At the same time as the prompt appears, a window in the lower right hand corner of the screen gives a graphical display of network canalization This shows a 2d representation of the network (irrespective of its native dimension) showing the degree of canalisation of each gene.

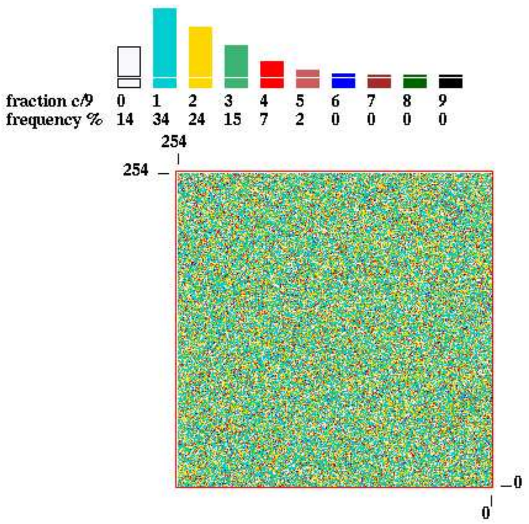
Above this is a histogram of canalizing saturation showing the frequency of different percentages of canalization, for 0%, 100%, and intervals of 25% in between, i.e. $0, < 25, < 50, < 75, < 100, 100$ (figure 15.2).



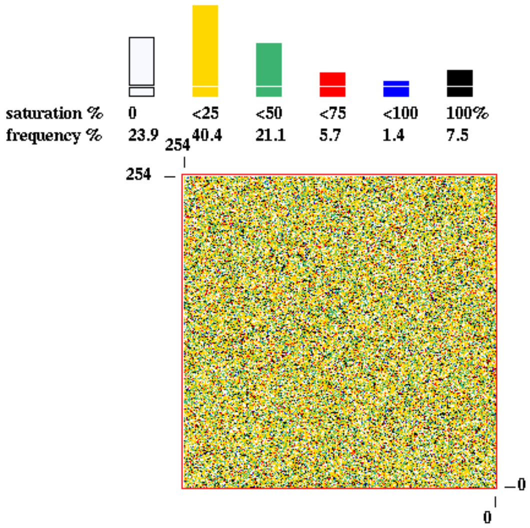
The initial k=5 genes with no canalizing inputs, as expected.

Canalizing inputs to the k=5 genes set to 45%.

Figure 15.3: Just one k in a mixed k network. Canalizing saturation for two networks with the same (roughly even) k-mix, 3-7, as shown in section 15.5. n=30×30. Only the selected k=5 genes are shown. An empty outline indicates no canalizing inputs and gaps are genes where $k \neq 5$. *left*: The initial network showing k=5 genes with no canalizing inputs, as expected. *right*: Canalizing inputs to the k=5 genes set to 45%. The colors in the network and histogram correspond.



A homogeneous k=9 network.



A mixed k network, k=3-13.

Figure 15.4: Examples of two large networks, $n = 255 \times 255$, with large k. The canalizing inputs were set to 20%. Several tries with small increments were required to reach this setting.

15.5.2 Canalizing for a particular k in a mixed k network

The canalizing setting of those genes in a mixed-k network with a just one specific k can be set in isolation from the rest of the network.

Enter the required value of k at the prompt **enter k for just one nhood, all-(def):** in section 15.5. It must be a valid value that exists in the k-mix.

For example, if k=5 is selected for the same network as shown in section 15.5.1, the following prompt will appear,

k=5 (19.1%) c-inputs=0/860=0.0% genes=0/172=0.0%)
set new % canalizing inputs, redo-q accept-ret:

This indicates that 19.1% of the network consists of k=5 genes, and that none of the inputs to those genes are canalizing, which is to be expected.

If, say, **45** (for 45%) is entered. The prompt will reapper with updated information,

k=5 (19.1%) c-inputs=387/860=45.0% genes=146/172=84.9%)
set new % canalizing inputs, redo-q accept-ret:

You may enter a new canalizing setting, or enter **return** to accept, the program will continue with options to review network architecture described in chapter 17. Enter **q** to revert to the first canalizing prompt (15.2).

At the same time as the prompt appears, a window in the lower right hand corner of the screen gives a graphical display of canalization (as in section 15.5.1) but for just the k=5 genes selected. An empty outline indicates no canalizing inputs and gaps are genes where $k \neq 5$.

Above this is a histogram of canalizing frequency for just the k=5 genes. Because we are dealing with just one k, the canalizing frequency is shown for $k \leq 9$, and the canalizing saturation for $k \geq 10$., as described in section 15.4,

15.6 Canalizing for large networks, or large k

For large networks, or large k, the algorithm for assigning canalizing may take some time. Also, the degree of canalization achieved may be less than requested. In this case more tries should be made (in sections 15.4, 15.5.1 and 15.5.2) to reach the required setting, increasing the setting by small increments.

A top center window monitors the percentage of canalizing inputs set so far. A progress bar near the top right hand corner of the screen also monitors the assignment of rules to the network, as in section 13.9

Chapter 16

Setting a singe rule

This chapter describes the alternative methods of setting a single rule, or single totalistic code. These methods also apply to setting rules (or codes) "by hand", for either a rule-mix (section 13.6) or a for a "limited subset of rules for rule selection" (section 14.3).

In most cases, except when loading a rule from a file, the methods apply equally to setting rules or totalistic codes, and can be used in combination.

16.1 The first single rule prompt

If one of the following selections was made at previous prompts,

return ... for a single rule in "Single rule or rule-mix", section 12.5,
h ... for "by hand" selection of a rule-mix, in "Methods for setting the rule-mix", section 13.3,
h ... for "by hand" selection of a rule-set in "Set the rule-set by hand or at random", section 14.1,

the following main sequence prompt is displayed (the precise wording is context dependent and may vary considerably) for example,

*for totalistic rules (**t** set in section 12.3)*
Select k5 totalistic code (def-dec) *(for example)*
maj-m rand-r bits-b hex-h repeat-p:

otherwise
Select k5 rule, empty-e fill-f maj-j Alt-A Life-L chain-c rand-r dec-d bits-b hex-h repeat-p load-l (def-rand): *(the default is context sensitive)*

16.2 Methods for setting a rule

A rule can be set in a variety of different ways,

empty-e ... reset all rule-table entries to 0, the program then reverts to the prompt in section 16.1. (not for totalistic codes).
fill-f ... reset all rule-table entries to 1, the program then reverts to the prompt in section 16.1. (not for totalistic codes).
after **f** *or* **e**, *the program reverts to the prompt in section 16.1.*
maj-m ... set the "majority" rule.
Alt-A ... set an "Altenberg" rule, where the output of each neighborhood, with density of 1s = d, is set to 1 with probability d. (not for totalistic codes).
Life-L ... set to the "game-of-Life" rule for a 2d $k = 9$ CA, or quasi "Life" otherwise. The prompt does not appear if $k < 5$, nor for totalistic codes).
chain-c ... a maximally chaotic rule, where $Z_{left} = 1$ or $Z_{right} = 1$, but not both. In addition, both Z_{left} and Z_{right} are greater than 0.5.
rand-r ... set the rule at random (the default), (the default is **d** for $k \leq 4$).
dec-d ... set the rule in decimal (for $k \leq 4$) (the default is **r** for $k \geq 5$).
bits-b ... set the rule as bits, "drawn" the rule-table on a 1d or 2d graphic array, using the mouse or keyboard.
hex-h ... set the rule in hex, in a mini "spread sheet".
repeat-p ... repeat the last rule that was set for the given neighborhood.
load-l ... load the rule from a `*.rul` file. (not for totalistic codes).

In most cases, except when loading a rule from a file, the methods apply equally to setting rules or totalistic codes.

Once set, the rule remains current and can be amended by resetting bits (bits-**b**), or resetting half bytes (hex-**h**). For example, after setting the "majority" rule, backtrack to the prompt in section 16.1 to modify it.

Resetting the rule to null (empty-**e**) allows a clean slate for setting the rule as bits or in hex. The various methods are described in detail below (they are similar to the methods for setting an initial state or seed in chapter 21.

16.3 Setting a single totalistic code

If **t** was selected in section 12.3, the single rule will be set as a totalistic code. Note that **empty-e**, **fill-f**, **Alt-A**, **Life-L chain-c** and **load-l** (in section 16.2) do not apply to totalistic codes, otherwise the methods for setting totalistic codes are equivalent to setting rules.

Once the totalistic code has been set, changes can be made to the rule itself. The rule-table bit pattern is displayed and can be amended as described in section 16.10.

16.4 Setting the majority rule

If **m** is selected, in section 16.1, the "majority" rule will be set. The majority rule outputs 1 for a majority of 1s in the neighborhood, 0 for a majority of 0s. In the case of even size neighborhoods, and a tie between 1s and 0s, the output is set according to the value at neighborhood index $k/2-1$

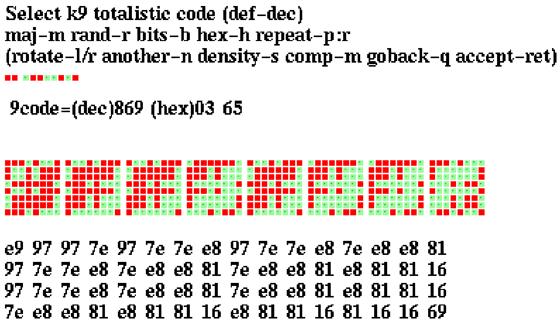


Figure 16.1: Setting a single totalistic code, k=9. An example of what appears on the screen. Once the code has been set, in this case at random, the rule-table bit pattern is displayed, and bits can be changed as described in section 16.10, then the hex expression of the rule is displayed.

(see chapter 9).

To modify particular bits in the "majority" rule, enter **q** to return to the prompt in 16.1, then select **b** to set bits in 16.10.

An alternative way of setting a majority rule, or any threshold rule, is by setting the rule as a totalistic rule (see totalistic code in section 12.2 and 12.3).

16.5 Setting Altenberg rules

not applicable to totalistic code

If **A** is selected, in section 16.1, the output of each neighborhood (with density of 1s = d) is set to 1 with probability d. This subset of rules produces interesting large scale structure in CA dynamics (suggested by Lee Altenberg).

16.6 Setting "game-of-Life" rule

applicable only for $k \geq 5$, not applicable to totalistic code

If **L** is selected in section 16.1 for a k=9, the rule for John Conway's "game-of-Life" [2] is set. The "life" rule is as follows, where a cell with value 1 is alive, and a cell with value 0 is dead,

game-of-Life rules
Birth: – A dead cell comes to life if its
neighborhood had exactly three live cells.
Death by overcrowding: – A live cell dies if its
neighborhood has 4 or more live cells apart from itself.
Death by exposure: – A live cell dies if its
neighborhood has less than 2 live cells apart from itself.

The "life" option is available for any $k \geq 5$ as well as the usual k=9 neighborhood, for 1d and 3d as well as 2d, and for non-local wiring and/or mixed neighborhoods. Note, however, that classical "life" behavior, characterized by the emergence of "gliders" and other interacting structures, occurs

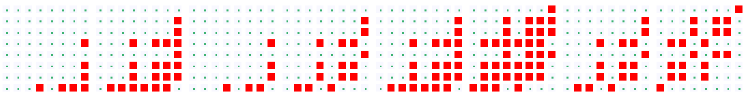


Figure 16.2: The game-of-life rule-table shown as a bit pattern. The $2^9 = 512$ neighborhood outputs are ordered according to neighborhood value, with the all=1s neighborhood top-left and all-0s bottom-right.

only for a 2d regular cellular automata where k=9, with a "Moore" neighborhood (as defined in figure 9.3, 9.4. Once selected, the "life" rule is displayed as a bit pattern, and in the rule window.

To make changes to particular bits in the "Life" rule, enter **q** to return to section 16.1 then select **b** to set bits in section 16.10.

16.7 Setting a chain rule

Chain rules are maximally chaotic rules, where $Z_{left} = 1$ or $Z_{right} = 1$, but not both. In addition, both Z_{left} and Z_{right} are greater than 0.5 to ensure minimum structure in space-time patterns. The typical dynamics of chain rules have extreemly low convergence, states usualy have just one pre-image making a "chain", and the garden-of-Eden density becomes order zero with increasing system size. Chain rules are suitable for encryption, for example, by running the information backwards to encrypt, forwards to decrypt.

16.8 Setting the rule in decimal

applicable only for $k \leq 5$

If **d** is selected in section 16.1, the rule can be specified by its decimal equivalent. The following prompt is displayed,

k=3 rule, enter 0-255 (def-rand): *(for example)*

If just **return** is entered, or a number outside the permitted range, the selection will revert to **random** as in section 16.9 below. When selected, the rule is also shown in hex and as the rule-table bit pattern.

16.9 Setting the rule at random

If **r** is selected in section 16.1, a random rule is generated and displayed as a rule-table bit pattern, and also in decimal (for $k \leq 5$), and hex. Details of the rule's λ and Z parameters, canalizing inputs, and the current density bias (λ bias) are also shown in top right windows.

A rule can be accepted or further alternative random rules may be generated indefinitely. The rule-table can also complimented, and the probability of setting 1s (i.e. tuning the λ parameter) can be altered (the default is 50%). The rule-table can be progressively mutated to force the

Z-parameter up or down. In addition the rule-table can be "rotated" to the left or right as if it were a periodic 1d state. The following prompt is displayed,

(rotate-l/r another-n density-s Z-u/d comp-m goback-q accept-ret)

Enter **return** to accept the random rule. Enter **n** for another random rule.

16.9.1 Rotating the random rule

If **l** or **r** is entered in section 16.9 above, the rule-table bit pattern is transposed horizontally towards the left or right, with excess bits wrapping around.

16.9.2 Random rule with a given λ parameter

Enter **s** in section 16.9 to change the λ bias of the random rule, i.e. the density of 1s in the rule-table. A top right prompt will appear, for example,

enter % (def 50.000%, accept-ret strict-s:)

Enter the new probability of setting 1s (as a percentage), this becomes the new default. The λ bias will not necessarily give a rule with the exact λ. To achieve this enter **s** at the prompt above. The λ bias will then be applied "strictly", i.e. corresponding as precisely as possible to the value requested. The λ bias (and the "strictness" if set) will be maintained for further random rules generated with (**n** in section 16.9).

16.9.3 Forcing the Z-parameter up and down

Enter **u** or **d** (repeatedly) in section 16.9 to progressively force the Z-parameter up or down, by selectively mutating the rule-table, i.e. flipping bits at random positions, and only retaining the bit-flips that produce the desired change in Z. Data on the current state of the rule will be displayed as described in section 16.9.5 below.

16.9.4 Complimenting the random rule

Enter **m** in section 16.9 to transform the rule into its compliment, changing 1s to 0s and vice versa.

16.9.5 Random rule data

For each alternative rule, a top right windows gives various data about the rule. The lower window gives the actual density ot 1s, and the current λ bias. The upper window gives details of various rule parameters. In this example for a k=5 rule, a low λ bias was first set in section 16.9.2 above,

C=1/5=*3* zl=0.1875 zr=0.1875** *(values shown are examples)*
ld=0.09375 ld-r=0.1875 P=0.90625 Z=0.1875

This information is similar to that in the rule window described in section 16.14 and is decoded as follows,

C=1/5 … the number of canalizing inputs, in this case 1 out a possible 5 (k=5).
3** … shows exactly which of the k inputs are canalizing, in this case input 3 (if none this is not shown).
zl … Z_{left}.
zr … Z_{right}.
ld … the λ parameter.
ld-r … the λ ratio.
P … the P parameter.
Z … the Z parameter.

16.10 Setting the rule as bits

If **b** is selected in section 16.1, a bit pattern representing the current rule-table (initially just 0s) is displayed. For $k \leq 6$ this is a single row divided into byte size segments. Otherwise multiple rows are shown with a maximum of 32 rows (for k=13). The initial scale of the bit pattern depends on k, but this can be changed. The maximum rule-table index is in the top left hand corner, the zero index in the lower right hand corner of the grid. The current position on the grid is indicated in a top right inset window.

0s are coloured light green, 1s are coloured red. To provide a clean slate for setting 1 bits, reset the rule-table to all 0s in section 16 with **empty-e**, or conversely reset to all 1s with **fill-f** for setting 0s. Alternatively, use this bit setting method for fine adjustments to rule-tables set by other methods.

During the bit setting procedure, a top right reminder window displays the following,

keys: set bit h-1/0 v-2/9 move-arrows-move **rule index=15 rot=1**
mouse: move-click, draw-keep pressed *(the current rule-table index and rotation size)*
buttons: set 1-left, set 0-right
exp/contrect-e/c rotate-l/r/+/-, comp-m accept-ret

Bits in the rule-table are set to 1 or 0 with the mouse or keyboard. The bit pattern scale can be expanded and contracted, "rotated" left or right by preset amounts, and complimented. The methods, described below, are essentially the same as for setting bits for the initial network state (the seed) described in section 21.6.

16.10.1 Setting bits from the keyboard

The current bit to be updated is highlighted by a small green flashing cursor and its position is displayed in a top right inset window.

The left/right/up/down arrow keys are use to displace the cursor. The up/down arrow keys only apply for rules where $k \geq 7$, where the bits are presented in 2 to 32 rows. Press keys **1** and **0** (without **return**) to set bits to 1 or 0. This automatically moves the cursor one positionto the right, so horisontal lines of 1s and 0s can be drawn by holding down the keys. Vertical lines of 1s and 0s can also be drawn downwards, key **2** draws 1s and key **9** draws 0s.

The position of the keyboard cursor can also be moved with the mouse, which also draws/deletes bits (see section 16.10.2 below).

To accept the rule-table bit pattern, enter **return**. When selected, the rule is also displayed in decimal (for $k \leq 5$) and in hex, and in the rule window.

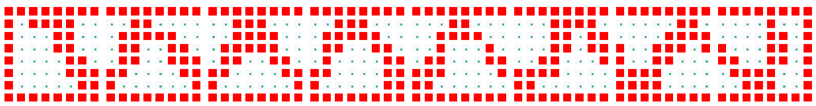
16.10.2 Setting bits with the mouse

While the cursor is within the bit pattern grid, the mouse can be used to draw 1s and 0s. There is a slight difference between DOS and Unix. In DOS the mouse cursor is confined within the bit pattern grid and just clicking the left or right mouse button sets single 1s and 0s. In Unix the cursor is not confined, but will change direction when within the grid, pointing north-west instead of north-east. Within the grid, left or right mouse clicks reposition the small green flashing cursor for settin bits from the keyboard, but do not set 1s or 0s.

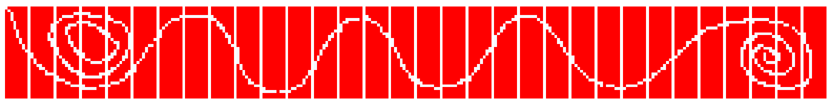
In both cases dragging the mouse with the left button depressed draws bits (i.e. sets 1s), and with the right button depressed deletes bits (i.e. sets 0s). Note that in Unix the centre button does not play a role. While a left or right button is depressed, a reminder is shown in the small top right inset window, for example,

left button: set 1s or **right button: set 0s**

To accept the rule-table bit pattern, enter "return". When selected, the rule is also displayed in decimal (for $k \leq 5$) and in hex, and in the rule window.



A k=9 rule-table, with 2^9=512 bits arranged in 8 rows of 64 bits, were each row is divided into 8 bit segments. Top left and lower right bits correspond to rule-table index 511 and 0.



A k=13 rule-table, with 2^13=8192 bits arranged in 32 rows of 256 bits, were each row is divided into 8 bit segments. Top left and lower right bits correspond to rule-table index 8191 and 0. In this example all bits were first set to 1 with **fill-f** in section 16, then 0 were drawn with the right mouse button.

Figure 16.3: Setting a rule as bits with the mouse or keyboard.

16.10.3 Expanding or contracting the rule-table bit pattern

Enter **e** or **c** in section 16.10 to expand or contract the scale of the rule-table bit pattern.

16.10.4 Rotating the rule-table bit pattern

Enter **l** or **r** in section 16.10 to "rotate" the rule-table, transposed the bits horizontaly by the default rotation setting (initialy 1 bit) towards the left or right, with excess bits wrapping around, as in section 16.9.1. To increase or decrease the default rotation setting enter + or **-**. This will be reflected in the top right indet window, for example **rule index=458 rot=6**.

16.10.5 Complimenting the rule-table bit pattern

Enter **m** in section refSetting the rule as bits to to transform the rule into its compliment, changing 1s to 0s and vice versa, as in section 16.9.4.

16.11 Setting the rule in hex

```
ff ff ff fe ff fe fe e8 ff fe fe e8 fe e8 e8 80
ff fe fe e8 fe e8 e8 80 fe e8 e8 80 e8 80 80 00
ff fe fe e8 fe e8 e8 80 fe e8 e8 80 e8 80 80 00
fe e8 e8 80 e8 80 80 00 e8 80 80 00 80 00 00 00
```

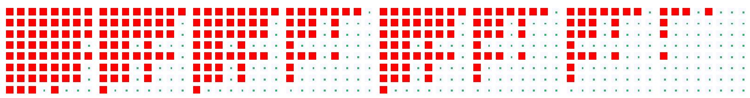


Figure 16.4: Setting the rule in hex. In this example the hex representation of the k=9 "majority rule" is shown, with its bit pattern representation below.

If **h** is selected in section 16, the hexadecimal (hex) representation (initially just 0s) of the current rule is displayed. Each hex character shows the value of 4 bits. The hex display is divided into bytes, pairs of hex characters. The hex character to be updated is highlighted by a green flashing cursor, which can be moved with the arrow keys.

During the hex setting procedure, a top right reminder window displays the following,

enter hex, arrows-move byte count=45 *(the current byte position from the left)*
rotate-l/r, accept-ret

The left/right/up/down arrow keys are use to displace the cursor. The up/down arrow keys only apply for rules where $k \geq 8$, where the hex digits are presented in 2 or more rows. To overwrite, enter a hex character (**1-9** or **a-f**) from the keyboard, without **return**. This automatically moves the cursor one position to the right. The current byte position on the hex grid (from the left) is indicated in a top right inset window, where the top right byte is at at position 0.

The hex rule-table can be rotated (by one bit) in the same way as in sections 16.9.1 and 16.10.4. To accept the hex expression of the rule, enter **return**. Once selected, the rule is also displayed in decimal (for $n \leq 5$), as the rule-table bit pattern, and in the rule window.

16.12 Repeating the last rule

If **p** is selected in section 16, the last rule that was set for the given value of k is repeated by automatically loading a `last[`k`].rul` file (for example, `last5.rul` for k=5). The file would have been automatically saved when the last rule with that particular k was selected (see section 16.15.

In the case of a rule-mix where a sequence of rules is entered by hand (see section 13.6 and 14.3), the previously entered rule is repeated. In the case of a k-mix the last rule with the appropriate k is loaded (for a totalistic code a random code is assigned). The repeated rule is displayed in decimal (for n¡=5), in hex and as the rule-table bit pattern, and in the rule window.

16.13 Loading a single rule

not applicable to totalistic code

If **l** is selected in 16, top right filing windows are presented allowing a rule to be loaded from a `.rul` file, the default is `myrule.rul` (see Filing, chapter 35). If successfully loaded, the rule-table bit pattern is displayed with the following message,

rule loaded, revise-q, cont-ret:

16.13.1 Single rule file encoding

The binary file defining the rule is encoded as follows: Byte 0 = k, the rest of the rule-table (from 0 to n-1) is set as bits in successive bytes.

16.13.2 Conflicts in the the size of k between network and file

For a single rule network, if the file to be loaded is for a different k than was specified for the network, the file will be accepted (in the main prompt sequence only, not when resetting rules at a later stage). The k setting of the network will be automatically changed to correspond to the file k, A top left message will confirn that this has occured, for example,

rule loaded, k changed from 5 to 9, cont-ret: *(values shown are examples)*

Once loaded, the rule is displayed in the rule window.

For a network with a rule-mix (or when resetting a single rule at a later stage), a file with a different k will not be accepted, with a top left message, for example,

not loaded, tried k=9, this k=3, cont-ret: *(values shown are examples)*

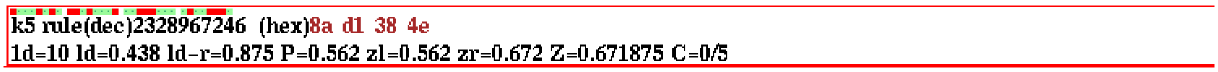
16.14 The rule window

When a rule is finally set, it is displayed in a lower rule window. The window shows the rule as a bit pattern (for $k \leq 6$), in decimal (for $k \leq 5$), and in hex. For $k \leq 11$ the full hex rule will not fit in the rule window; as much as will fit will be shown.

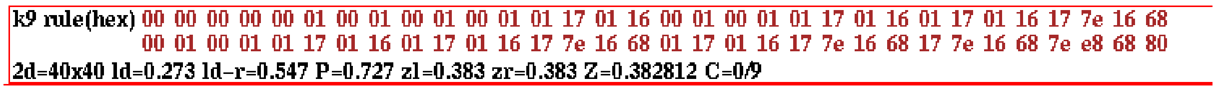
The window also gives the network size, and details about the rules λ and Z parameters, and canalizing inputs. This window is updated as rules are transformed or mutated. An example of the data in a rule window is shown below,

■·■■■·■· *(the look-up table shown as a bit pattern)*
k**3 rule=(dec)186 (hex)ba** *(values shown are examples)*
1d=10 ld=0.625 ld-r=0.75 P=0.625 zl=0.75 zr=0.25 Z=0.75 C=1/3 **0

	key to data in the rule window
k3-rule ...	the neighborhood size k, the rule number in decimal and hex.
1d=10 ...	the network dimention and size for 1d. or the equivalent for 2d or 3d, **2d=40x40** or **3d=20x20x20**, for example.
ld ...	the λ parameter.
ld-r ...	the λ ratio.
P ...	the P parameter.
zl ...	Z_{left}.
zr ...	Z_{right}.
Z ...	the Z parameter.
C=1/3 ...	the number of canalizing inputs, in this case 1 out a possible 3 (k=3).
****0** ...	shows exactly which of the k inputs are canalizing (if none this is not shown), in this example there is 1 canalizing input, at neighborhood index 0.



a k=5 rule.



the k=9 game-of-Life rule.

the k=13 majority rule, only the start of the hex rule fits in the window.

Figure 16.5: Rule window examples.

16.15 Automatic saving of last rule

After a single rule has been set, it will be automatically saved with a default name `last[`k`].rul` depending on k. For example, if the k=5 the file name will be `last5.rul`.

16.16 Transforming the single rule

Once the single rule has been set, a top right prompt allows it to be transformed in various ways. The following top right prompt is presented,

transform rule: solid-o invert-v comp-c neg-n ref-r, canalizing-C:
equiv greater k (4-13), eff k: all-k, save-s: *(if k=3)*

The various methods for transforming rules are described in chapter 18, and the prompt may also be accessed from the wiring graphic prompt in chapter 17 .

Chapter 17

Reviewing network architecture

This chapter describes ways of reviewing the network architecture by the following methods,

wiring matrix ... shows the wiring (and rules if set) in a table or spread-sheet format, where the wiring can be changed.

1d, 2d or 3d wiring graphic ... the most powerful method for tailoring the wiring (and rules if set), and also seeing the details of the network.

Alternativly the network can be reviewed as a graph, described in chapter 20. The graph option does not allow changes to the underlying network, but the graph can be unraveled by dragging nodes and components. The graph can be rotated, expanded, contracted, and many other manipulations performed. Default presentations include: circle, spiral, 2d and 3d.

The methods described in this chapter, the wiring matrix and wiring graphic, allow wiring and rules, set in chapters 9 - 12, to be examined, changed, and tailored to requirements, including biased random settings to pre-defined parts of the network. These are very flexible methods, and for RBN its usualy easier to set up a suitable dummy network initially, then tailor it here.

Although there are some differences in the methods for amending the network between the 1d, 2d and 3d wiring graphic formats, in most cases they are the same or similar.

The 1d graphic has two alternatives. Wiring can be shown between successive time-steps, or between cells arranged in a circle. Both 1d methods apply whatever the native dimensions of the network. The 2d graphic can be applied to both 1d and 2d networks. For a 3d network, the wiring graphic shows a simultaneous display in both 2d and 3d (2d+3d), where the 2d graphic shows horizontal slices (levels) through the 3d network stacked above each other.

17.1 The network architecture prompt

Once special wiring (chapter 11), or both the wiring and rules, have been set, a network architecture prompt is presented in the top right corner of the screen. The exact wording and options offered depends on other settings. The prompt can also be activated at later stages in DDLab, while space-time patterns (section 32.14) or attractor basins (sections 30.4, 30.5, 30.5.1) are being drawn.

The prompt starts as as follows, where the network sizes shown are examples,

1d network (n=150), ... *(for a 1d network)*
2d network (40x40), ... *(for a 2d network)*
3d network (15x15x15), ... *(for a 3d network)*

The first line of the prompt continues,

... **wiring only - rules not set** *(if rules have not been set)*
... **review/revise/learn, wiring and rules** *(if rules were set)*

The prompt continues as follows,

graph-g, matrix: rev-m view-M:
graphic: 1d+timesteps-1 circle-c, 2d-2: (*or* **2d+3d-3:** *for a 3d network*)

For example, for a 1d network with rules set the prompt appears as follows,

1d network (n=150),review/revise/learn, wiring and rules
graph-g, matrix: revise-m view-M
graphic: 1d:timesteps-1 circle-c, 2d-2:

17.2 Setting the "wiring matrix"

Enter **m** or **M** in section 17.1 to show the the network architecture in a spread-sheet type format, the "wiring matrix", (see also section 11.6, "Wiring by hand"). Enter **M** to just view the network, **m** to also make changes to the wiring. The rules (if set) are also shown, but can not be changed (this can be done in the "wiring graphic", section 17.3 below).

Examples are shown in figure 17.1. The rules are listed in hex, but for large k only part of the hex rule will fit in the window. The rule for the cell index currently active appears in the rule window described in section 16.14.

17.2.1 The wiring matrix

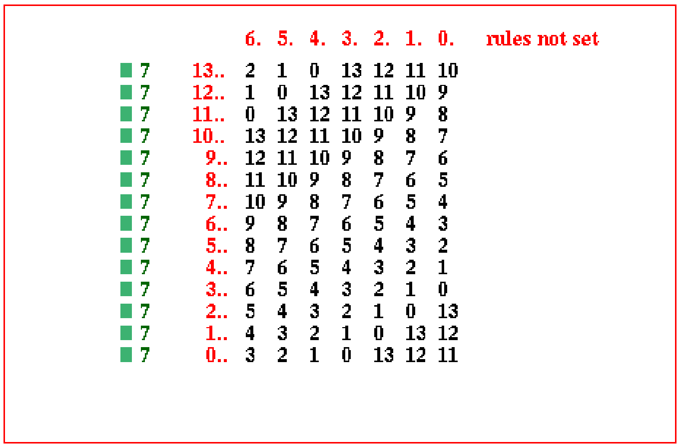
As described in section 11.6, columns give the cell's pseudo-neighborhood index, K (k-1...0). Rows give the cell network position, N (n-1...0). The 0-0 grid (or the 0-minimum n grid if the whole matrix does not fit within one window) is in the lower right hand corner. Each grid records the position in the network (n-1...0), to which the K'th wire of the N'th cell is connected.

A column to the left of the cell index (colored green) shows the out-degree (and out degree histogram) of each cell.

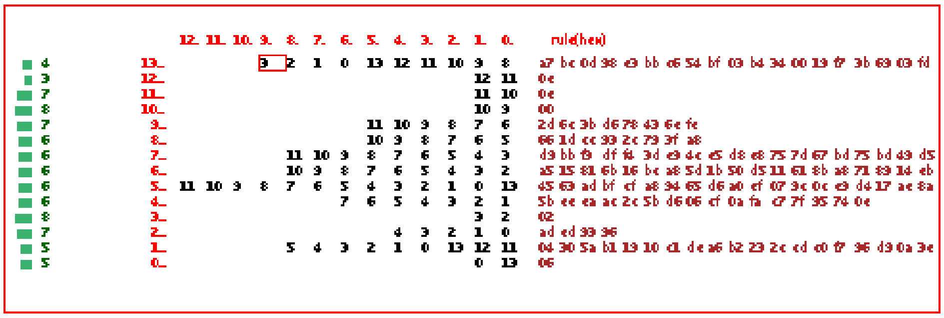
17.2.2 Showing the matrix

If **M** was selected in section 17.1, the wiring matrix is simply displayed, together with the following top right prompt,

(if the matrix fits one window)
print-p quit-q layout=l font-f jump to index (1599-0): (**print-p** *for DOS only)*
(for a large matrix requiring a succession of windows, this example for a 2d network, 40x40)
more-ret print-p quit-q layout-l font-f jump to index (1599-0):



The wiring matrix of a 1d CA with periodic boundary conditions. k=7, n=14, rules not set.



The wiring matrix for a mixed k network with random wiring. n=14, k=1-13, with rules set.

Figure 17.1: Wiring matrix examples. k-1...0, indexes columns, n-1...0, indexes rows. The column on the left shows the "out-degree" of each cell, the number of output wires that link to it, also shown as a histogram. If rules have been set, they are shown in hex (as much as will fit) on the right, in the column "rule(hex)" (otherwise "rules not set" is shown). The rule of the active cell also appears in the rule window described in section 16.14.

Enter **l** or **f** to alter the presentation of data and the amount visible in the matrix window. **f** toggles the font size between normal and small. **l** changes the matrix window width. The following top right prompt is presented,

> **change window width (922-231 now=462):** *(values shown are examples)*

Enter the new width in pixels within the limits indicated.

Enter **return** to see further sections of the matrix in successive windows (for large networks), or to exit the matrix and revert to the prompt in section 17.1. Enter **p** to print the current matrix (DOS only). Enter **q** to revert to the prompt in section 17.1. Enter the index position to jump to a new cell index, which will become the first entry in the top row.

17.2.3 Amending the matrix

If **m** was selected in section 17.1 the matrix is displayed and can also be amended.

The following top right reminder is presented,

hand wire/revise:jump-j *(for a 2d network, 40×40)*
enter wiring positions 0-1599 (return on blank/0=random
move-arrows more/complete-m layout-l font-f quit-q

A green flashing cursor appears over the active cell/wiring entry, which is also outlined. The outline remains for all entries visited. Move around the matrix with the left/right/up/down arrow keys. To change, enter the new position at the flashing cursor and complete the entry by moving to another grid with an arrow key or entering **return**. On a blank grid (see section 11.6), or 0, **return** gives a random position. A new position outside the network limits will be ignored.

Enter **l** or **f** to alter the presentation of data and the amount visible in the matrix window. **f** toggles the font size between normal and small. **l** changes the matrix window width, as described in section 17.2.2 above. Large networks may require several successive windows to display the matrix. Enter **m** to see the next window. Enter **j** to jump to a new cell index, the following prompt is presented,

jump to index (1599-0): *(for a 2d network, 40×40)*

Enter the new cell index, which will become the first entry in the top row. Moving outside the first or last entry in the current window with the arrow keys or return also brings up new window.

Enter **q** to accept the wiring and revert to the prompt in section 17.1.

17.3 Selecting the "wiring graphic"

Enter **1**, **c**, **2** or **3** in section 17.1 to show the network as a graphic diagram, where both wiring and rules can be viewed amended or reset by very flexible methods, described below. The 1d graphic has two alternatives, wiring shown between successive time-steps (enter **1**), or between cells arranged in a circle (enter **c**). Both alternatives apply whatever the native dimensions of the network. The 2d wiring graphic may also be selected for a 1d network. The 2d+3d wiring graphic only applies to 3d networks. The display and amendments apply to the "active cell" or to a pre-defined block of cells.

17.4 Wiring graphic, 1d

Enter **1** or **c** in section 17.1 for a 1d graphic. This shows how a particular cell (the "active cell"), or a pre-defined 1d block of cells, is wired in a 1d, 2d or 3d network.

If **1** is entered, the wiring is shown between two time-steps, from time-step t_0 to t_1, as in figure 17.2.3 and elsewhere. Network cells at t_0 are shown in a row above cells at t_1, and the out-degree of each cell is indicated by the height of the cell's representation at t_0 (the cell wiring out-degree histogram).

If **c** is entered, the wiring is shown between cells, represented as nodess or radial lines, arranged in a circle, as in 17.2.3 and elsewhere. The zero index is due east and increases clockwise. The cell index is shown within each disk for networks smaller than about $n = 40$. For networks greater than about $n = 162$, cells are shown as short radial lines. Each cell's "outwires" are represented by the length of radial lines outside each cell (the cell wiring out-degree histogram).

In both presentations, an option allows the pseudo-neighborhood to be omitted, showing just direct links. The active cell is moved around the network with the arrow keys, and its possible to "jump" to a new location. The rule for the active cell appears in the rule window described in section 16.14.

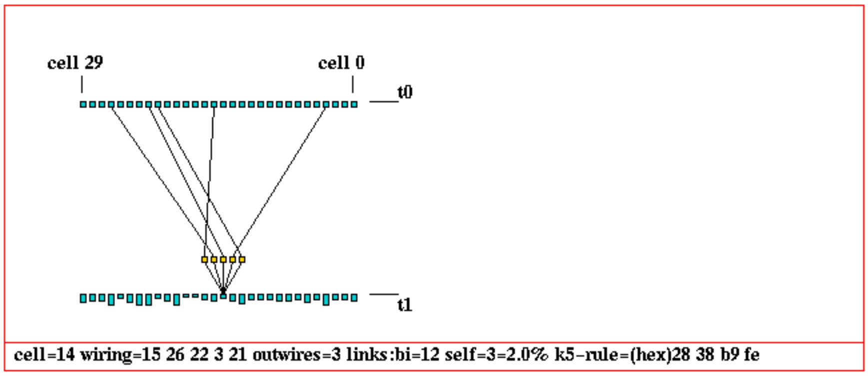


Figure 17.2: The 1d wiring graphic showing the wiring between successive time-steps, $k = 5$, $n = 30$. Each cell's "outwires" are represented by the height of the lower cells, at time-step t_0.

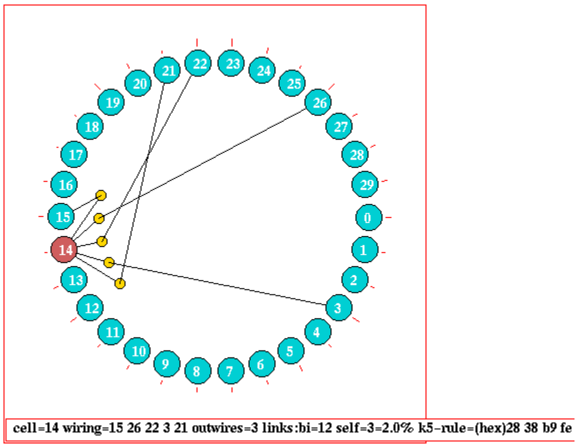


Figure 17.3: The 1d wiring graphic showing the wiring between cells arranged on a circle, the same network as shown in figure 17.2.3, $k = 5$, $n = 30$. The cell index is shown for networks smaller than about $n = 40$. For network greater than about $n = 162$, cells are no longer shown as nodess, but a short radial lines. Each cell's "outwires" are represented by the length of red radial lines outside each cell.

17.4.1 Data - 1d wiring graphic

Various data about the active cell's wiring are shown below the 1d graphic For example, in figures 17.2.3 and 17.2.3, a $k = 5$ network, with mixed rules and non-local wiring, the data is as follows,

cell=14 wiring=15 26 22 3 21 outwires=3 links:bi=12 self=3=2.0% k5-rule=(hex)28 38 b9 fe

See section 17.7 to decode this data. The rule and rule data for the active cell are also shown, if the rule/s have been set, in the rule window described in section 16.14.

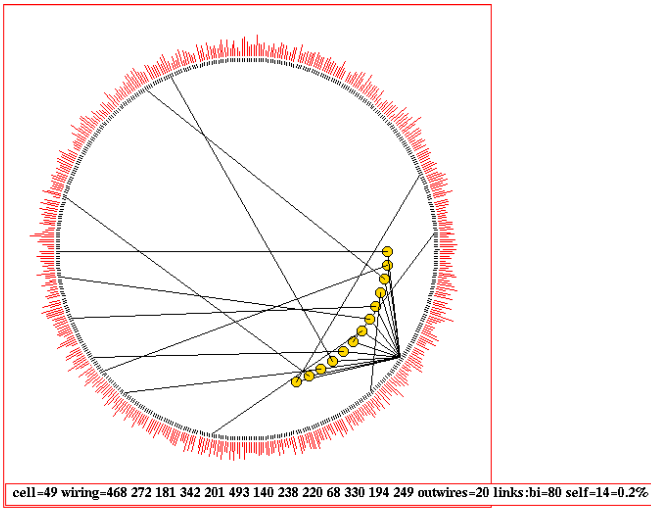
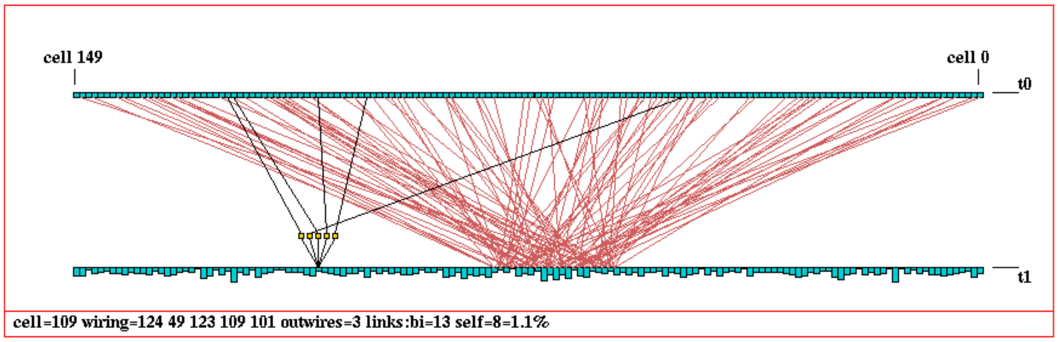


Figure 17.4: The 1d wiring graphic showing the wiring between cells arranged on a circle. $k = 13$, $n = 500$. Cells are shown as short radial lines (for networks greater than about $n = 162$). Each cell's "outwires" are represented by the length of red radial lines outside each cell.



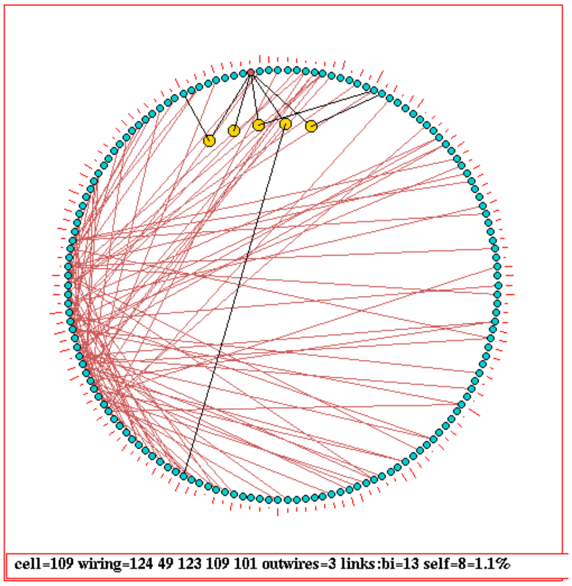


Figure 17.5: The 1d wiring graphic, showing wiring to a block. *Above*: As time-steps. *Left*: As a circle. k=5, $n = 150$. The block was defined from cell 60-80. The "active cell" (109) is still visible, and can be moved as usual.

17.4.2 Wiring graphic prompts, 1d

The following top right prompts/reminders appear at the same time as the 1d graphic. Various context dependent options are presented for analyzing the network, and amending the wiring or rule/s for a single cell, a group of cells, or for the network as a whole. For 1d mixed k networks cells can also be deleted, shortening te network. An example of the prompt is as follows,

> **1d (n=150) wiring/rules: move-arrows jump-j**
> **one cell: right/left arrows, 15 cells: up/down arrows**
> **block-b tog:all-g pseudo-p, in/out-i/o avZ-z routine learn-l**
> **rewire 1d cell 74: untangle-u hand-h rand-r/R special-s local:1d-1**
> **change k=8(max 13)-k, kill-K del-d, rule:rev/trans-v/t, Derrida plot-D**
> **file/data-f hist:In(k)/Out/Both-I/O/B, reset-q cont-ret:**
> *(Note:* **-z** *for mixed rules,* **-z -l -v/t -D -f** *if rules set,* **-k** *if rules not set or mixed rules,*
> **-K -d** *if rules set and mixed-k,* **-d** *in main prompt sequence only)*

If the rules have not been set, the first line of the prompt reads,

> ... **wiring only** *(instead of ...* **wiring/rules***)*

These options activate as soon as the key is pressed, without pressing return. Below is just a brief summary, more details are given in further sections in this chapter.

move-arrows ...	use the arrow keys to move around the network.
jump-j ...	to jump to a particular cell index.
block-b ...	define a block of cells (in 1d, 2d or 3d), which can be the whole network.
tog-g ...	toggle to show all the wiring (not 2d or 3d presentation).
pseudo-p ...	toggle the "pseudo neighbourhood".
in/out-i/o ...	show the indirect inputs and outputs ("time-steps" or "circle" presentation only).
avZ-z ...	compute the (weighted) average λ and Z for a mixed rule network.
learn-l ...	implement the learning/forgetting routine.
untangle-u ...	"untangle" the cell's wiring.
hand-h ...	rewire the cell by hand.
rand-r/R ...	randomly rewire with a bias according k in mixed-k networks.
special-s ...	set "special" wiring biases, for rewiring with **r**.
local:1d-1 2d-2 3d-3 ...	set local CA wiring for the cell or block.
change k=8(max 13)-k ...	change k for the cell or block up to max-k.
kill-K ...	kill or neutralize a cell by cutting all its links, both inputs and outputs, except for one input to itself. The k=1 rule is set to 2, meaning its value stays constant, with effective k=0.
del-d ...	delete a cell from the network, and shorten the network by one.
rule:rev-v ...	revise the rule for a cell, and copy to a block.
rule:trans-t ...	transform the rule or rules, set Canalizing inputs.
Derrida plot-D ...	draw the Derrida plot for the network.
file/data-f ...	filing, save and load the network architecture.

hist:In(k)/Out/Both-I/O/B ... show a histogram of the frequency distribution of inputs (i.e. k), outputs, or both (i.e all connections) in the network. distribution.

reset-q ... backtrack to redisplay the network (section 17.1).

Options **z,l,h,r,R,s,1,2,3,k,K,v,t,D,f,I,O,B** are described jointly for 1d, 2d and 3d networks, in section 17.8. Options **j,b,g,p,i,o,u,d** apply more specifically, or uniquely, to the 1d graphic, and are described below.

17.4.3 Moving or jumping between cells, 1d

To move the active cell, use the arrow keys. The left and right arrow keys move by one cell, the up and down arrow keys move by one tenth the network size left and right, for sizes of 20 and over. Alternatively, enter **j** to jump to a specific cell index. The following prompt is displayed,

jump to index (1295-0): *(this example for a network size 1295)*

Enter the cell index required.

17.4.4 Defining a block, 1d

Enter **b** in section 17.4.2 to define a block of cells. The following top right prompt is presented,

1d block-1, all-a single cell-def: *(then, if* **1** *is selected...)*
1d block (0-49, def 102), low: high:

Enter **1** to define the block, **a** to define the whole network as a block. If **1** was entered, enter the low and high cell indices delimiting the block. The wiring of all the cells in the block will be shown in the graphic as direct links, colored light red. The active cell will still be visible and can be moved as usual. For a 1d network shown in 2d, the prompt is as described in section 17.5.4

An active block is indicated in the wiring graphic prompts (17.4.2) for example **rewire 1d block 22-33:**. To deactivate the block enter **b** in section 17.4.2, then **return**.

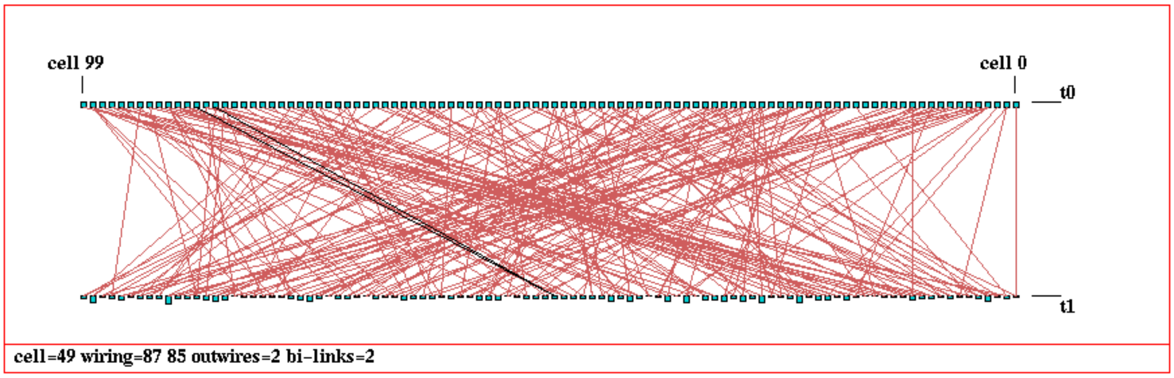
If the block is active the following options will apply to the whole block, for example,

...
rewire 1d block 60-80: untangle-u hand-h rand-r special-s local:1d-1
change k=8 (max 13)-k rule:rev/trans-v/t
...

The rule options actually apply to the active cell, but it can then be copied to the bloc.

17.4.5 Toggling the block, 1d

Enter **g** in section 17.4.2 to toggle the block defined in section 17.4.4, making it either visible or invisible, without deactivating it. If this is done when a block is not active, the wiring in the whole network will be toggled.



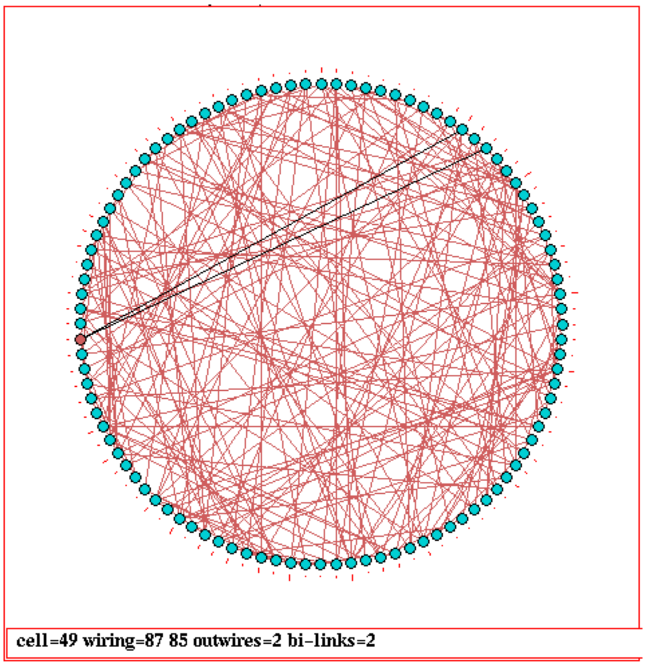


Figure 17.6: The 1d wiring graphic, showing the wiring of the whole network. *Above*: As time-steps. *Left*: As a circle. k=2, n = 100. If a block is not set in section 17.4.4, entering **g** in section 17.4.2 toggles between the wiring of the active cell and the whole network.

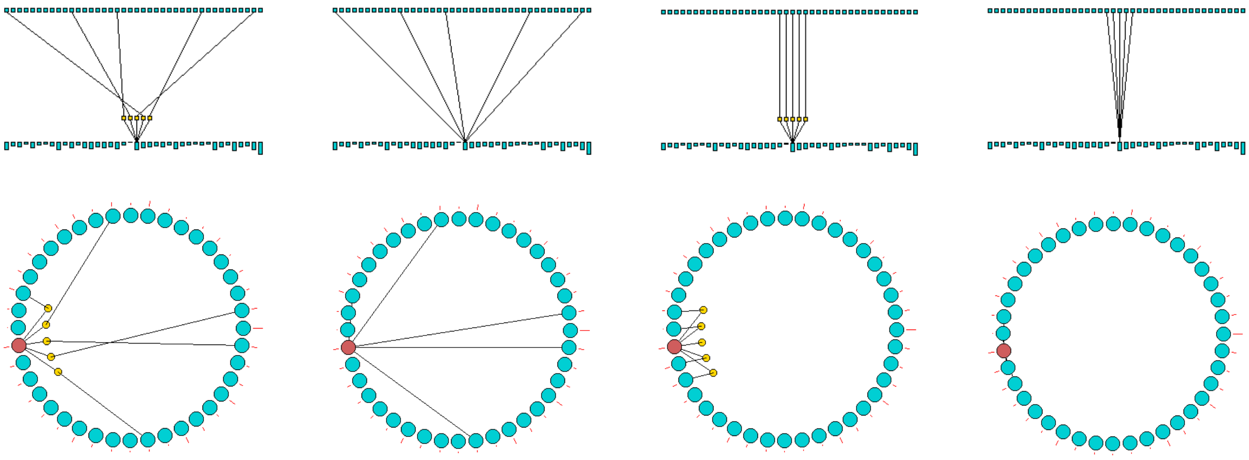
17.4.6 Include the pseudo-neighborhood, or direct wiring only, 1d

The default wiring representation for the active cell includes the pseudo-neighborhood. Alternatively just the direct connections may be shown, always the case for a block. Enter **p** at the prompt in section 17.4.2 to toggle between the two.

17.4.7 Recursive inputs to a cell, 1d

Enter **i** in section 17.4.2 to show all recursive inputs to a cell, whether direct or indirect, showing the "degrees of separation" between cells. This will include inputs to inputs, and so on, relative to past time-steps, showing the potential *upstream* influence on the cell from past network dynamics.

Initially, direct wiring (see section 17.4.6) will be shown from the previous time-step in the usual way. The following prompt is displayed in the top left hand corner of the wiring window,



a. random wiring pseudo-n'hood | b. random wiring direct | c. CA wiring pseudo-n'hood | d. CA wiring direct

Figure 17.7: Examples of the 1d wiring graphic, including the pseudo-neighborhood, or just direct wiring, k=5, $n = 40$. *Above*: As time-steps. *Left*: As a circle.

step 1, inputs=2/100, step-s all-def: *(for a k=2, n=100 network)*

Enter **return** to generate all inputs in one go. Enter **s** to show the inputs in steps, by entering **return**, or enter **q** to quit early. In both cases, inputs that relate to different past time-steps are shown in different colors (cycling through about 7 colors). For networks smaller than ($n = 1500$), a small 2d insert in the top right hand corner of the wiring window indicates the fraction of cells connected so far. Intermediate prompts show the number of inputs so far, for example,

step 4, inputs 27/100=27% cont-ret: *(values shown are examples)*

Once all possible input cells have been found, the following prompt appears in the top left hand corner of the wiring window,

step 12, inputs 80/100=80% complete *(values shown are examples)*

Enter **return** to reactivate the main wiring graphic prompt (section 17.4.2) and revert to the normal wiring graphic functions.

17.4.8 Recursive outputs from a cell, 1d

Enter **o** in section 17.4.2 to show all recursive outputs from a cell, whether direct or indirect, showing the "degrees of separation" between cells, the converse of of recursive inputs in section 17.4.7. This will include outputs from outputs, and so on, relative to future time-steps, showing the potential influence from the cell on future network dynamics *downstream*.

Initially just the immediate wiring outputs (if any) will be shown from t_0 to t_1. The number of these outputs may be less than that shown as "outwires" in the 1d wiring window data (see section 17.4.1) because there may be duplicate output wires which are only counted once. The following prompt is displayed in the top left hand corner of the wiring window,

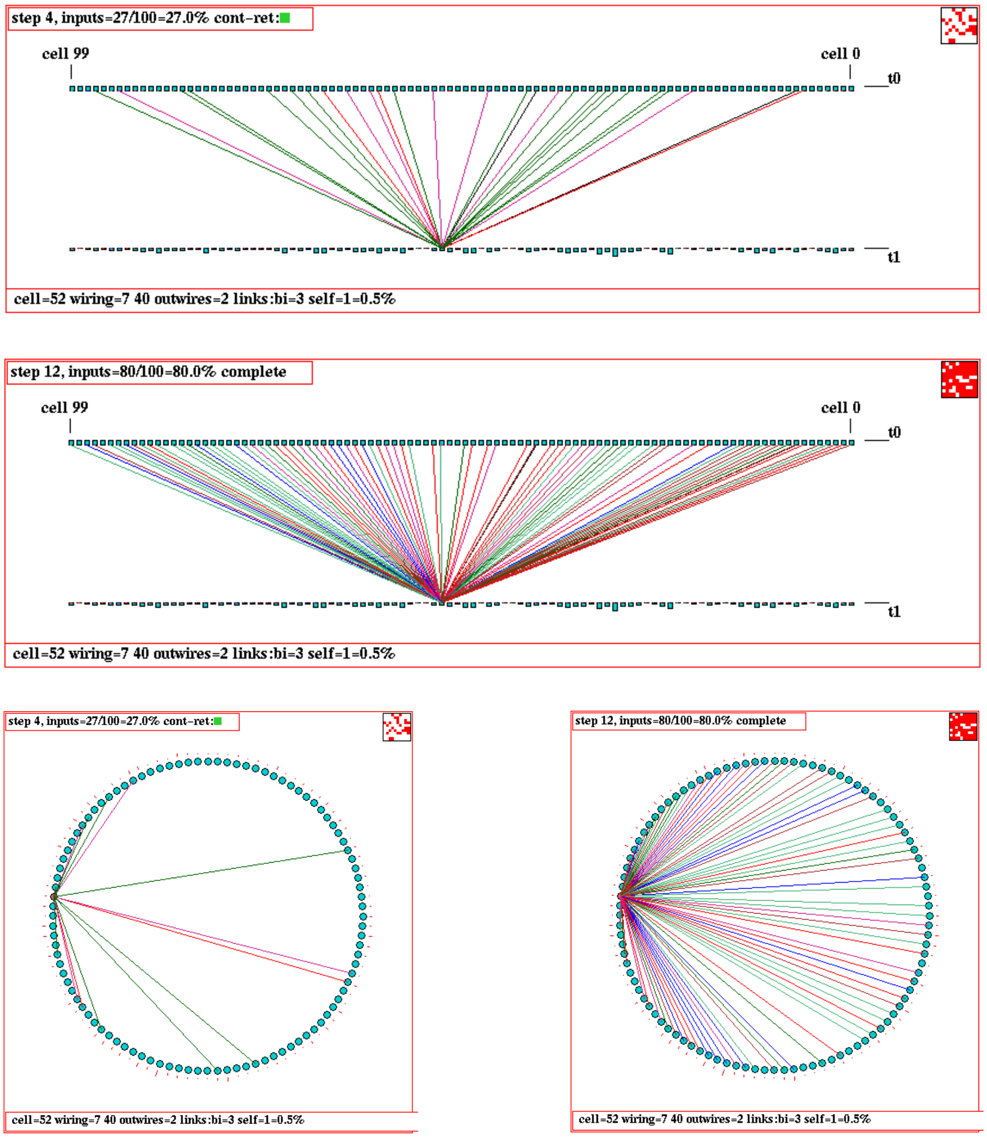


Figure 17.8: Recursive inputs (direct and indirect) to a given cell for a $k = 2$, $n = 100$ RBN. *Above*: As time-steps. *Below*: As a circle. Firstly, inputs are shown after 4 backward steps with 27 cells reached. Secondly, the complete inputs are shown after 12 backward steps with 80 cells reached. For networks smaller than $n = 1500$, a small 2d insert in the top right hand corner of the wiring window indicates the fraction of cells connected so far.

step 1, outputs=2/100, step-s all-def: *(for a k=2, n=100 network)*

Enter **return** to generate all outputs in one go. Enter **s** to show the outputs in steps, by entering **return**, or enter **q** to quit early. In both cases, outputs that relate to different future time-steps are shown in different colors (cycling through about 7 colors). For networks smaller than $n = 1500$, a small 2d graphic in the top right hand corner of the wiring window indicates the fraction of cells connected so far. Intermediate prompts show the number of inputs so far, for example,

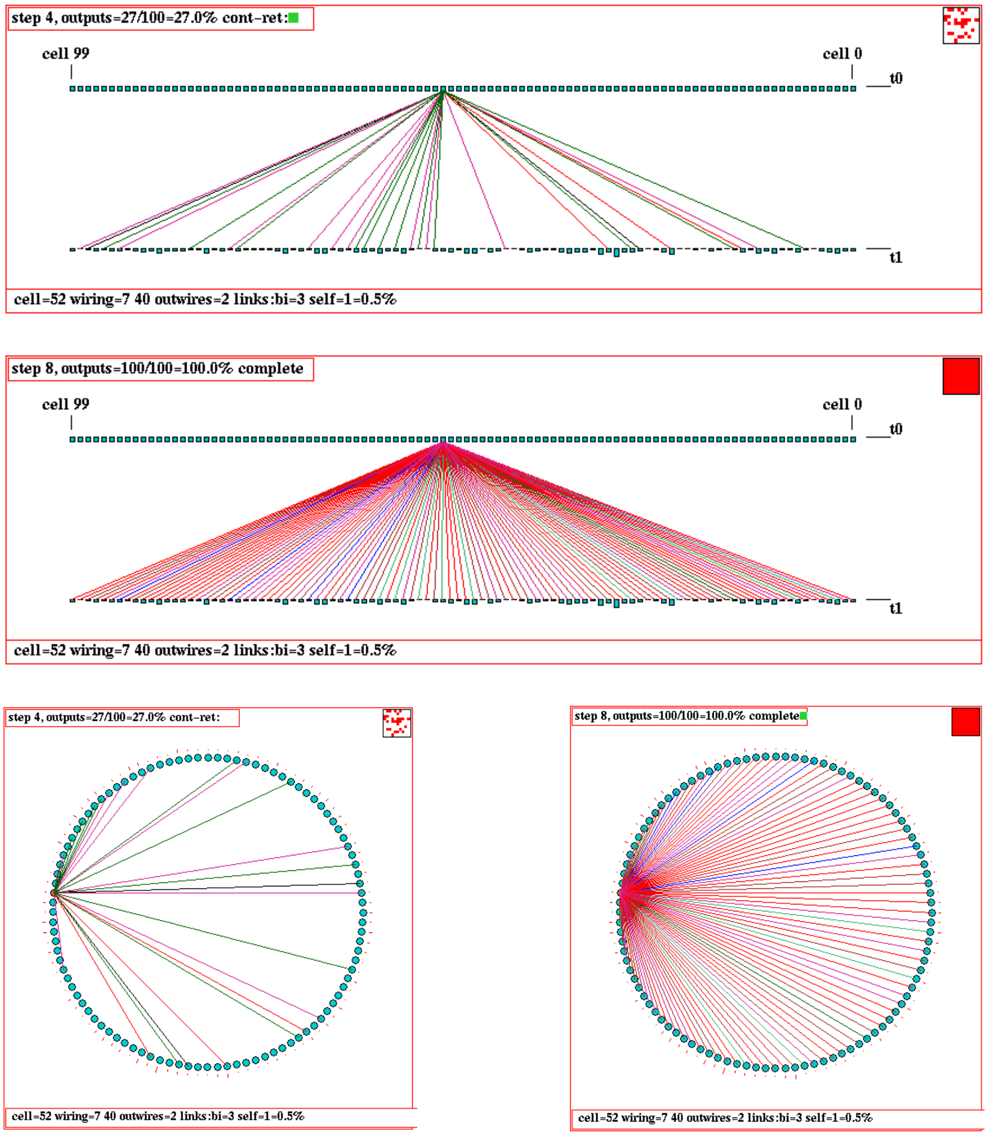


Figure 17.9: Recursive outputs (direct and indirect) from a given cell for a k=2, n=100 RBN. *Above*: As time-steps. *Below*: As a circle. Firstly, outputs are shown after 4 forward steps with 27 cells reached. Secondly, the complete outputs are shown after 8 forward steps with all 100 cells reached. For networks smaller than $n = 1500$, a small 2d insert in the top right hand corner of the wiring window indicates the fraction of cells connected so far.

step 4, outputs=27/100=27% cont-ret: *(values shown are examples)*

Once all possible output cells have been found, the following prompt appears in the top left hand corner of the wiring window,

step 8, outputs=100/100=100% complete *(values shown are examples)*

Enter **return** to reactivate the main wiring graphic prompt (17.4.2) and revert to the normal wiring graphic functions.

17.4.9 Untangling the wiring

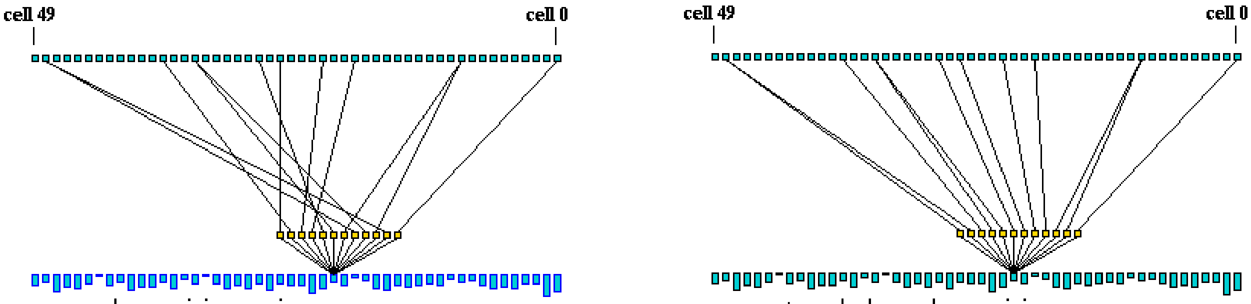


Figure 17.10: Untangling the wiring: inputs to the pseudo-neighborhood of each cell are rearranged so that their wires do not cross in the time-step representation, and the rule is automatically transformed to give equivalent behavior. In this example k=12, n=50.

Enter **u** in section 17.4.2 to "untangle" the wiring for the active cell, the defined block, or all cells in the network, so that wires do not cross each other in the 1d time-step representation (figure 17.10).

The following top right prompt is presented,

untange 1d wiring: all-a, single cell-(def): *(for the active cell)*
or
untange 1d wiring: block 60-80, all-a, single cell-(def): *(if a block is displayed)*

At the same time, rules (if set) are transformed for equivalent dynamics [1]. This only applies to a mixed rule network, not to a "single rule" network set in section 12.5. In this case the wiring will be untangled, but the rules will remain unchanged. However, a rule-mix where all the rule are the same can be set up as described in section 12.6.

17.4.10 Deleting a cell

Enter **d** in section 17.4.2 to delete a cell from the network and shorten the network by one cell. This option is only available in the main prompt sequence, not while interrupting space-time patterns or attractor basins. Backtrack to the main prompt sequence if necessary. In addition, the base network itself must be 1d and have mixed-k, with the rules set. When a cell is deleted, all links, both inputs and outputs, between the cell and other cells will be cut, and cell indices greater than the deleted cell will be reduced by one.

[1] If the order of the k inputs in the pseudo-neighborhood is changed, the rule can also be transformed to achieve identical behavior. Thus there are $k!$ equivalent pseudo-neighborhood orders and corresponding rules.

17.5 Wiring graphic, 2d

Enter **2** in section 17.1, for the 2d graphic. This shows how a particular cell (the "active cell"), or a pre-defined 2d block of cells, is wired in the 2d network (or in a 1d network presented in 2d). The 2d graphic can be expanded and contracted, and shifted up and down if necessary to see the relevant part. The active cell is moved around the network with the arrow keys, and its possible to "jump" to a new location. The rule for the active cell appears in the rule window described in section 16.14.

For 1d networks shown in 2d, the most reasonable i, j dimensions are automatically computed, with $i \geq j$ (for n prime $i = n, j = 1$).

The display can be toggled between 'direct wiring" and wiring to the pseudo-neighborhood (as in figure 17.4.10). By default, "direct wiring" is displayed if the wiring is non-local, and to the pseudo-neighborhood for local CA type wiring. There is also a 4-way toggle to alter the presentation of connections, and the cells connected, which is especially useful in visualizing blocks.

The current cell is colored red, its pseudo-neighborhood yellow, the "actual neighborhood" cells are colored green (see examples in figure 11.3).

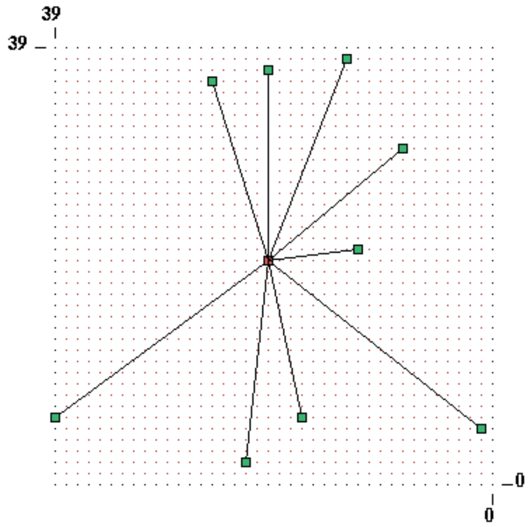


Figure 17.11: The 2d wiring graphic, for a 40×40 network, $k = 1 - 9$, max-k=13, showing direct wiring to a $k = 9$ cell. Wiring data at the foot of the graphic is shown in section 17.5.1.

17.5.1 Data - 2d wiring graphic

As for the 1d wiring graphic, various data about the active cell's wiring are shown at the foot of the 2d wiring graphic. For example, for figure 17.4.10 the data is as follows,

```
2d maxk=13 cell=20,20 k=9 wiring=7,37 29,20 23,37 6,25 5,11 26,37 27,2 9,27 0,5 outwires=6 links:total=11304 av-k=7.07 bi=21 self=4=0.0%
```

See section 17.7 to decode this data. The current cell position, and the actual neighborhood positions, are shown in the form i, j. The rule and rule data are shown in the rule window (section 16.14).

17.5.2 Wiring graphic prompts, 2d

The following top right prompts/reminders appear at the same time as the 2d graphic. Various context sensitive options are presented for amending the wiring or rule/s for a single cell, a 2d block of cells, or for the network as a whole.

The presentation options allow for expanding and contracting the scale of the 2d graphic, and if only part of the graphic fits within the display area, to move the network up and down to see the relevant parts.

2d (40x40) wiring/rules: move-arrows jump-j
exp/contr-e/c up/down/start-u/d/D
block-b tog:block-g links-n pseudo-p, avZ-z learn-l *(*-g *if block active)*
rewire 2d cell 26,24: hand-h rand-r/R special-s local:1d-1 2d-2
change k=8(max 13)-k, kill-K, rule:rev/trans-v/t, Derrida-D
file/data-f hist:In(k)/O/B-I/O/B, reset-q cont-ret:
(Note: **-z** *for mixed rules,* **-z -l -v/t -D -f** *if rules set,* **-k** *if rules not set or mixed rules,*
-K *if rules set and mixed-k)*

If the rules have not been set, the first line of the prompt reads,

... **- wiring only** *(instead of ...,* **wiring/rules***)*

Options **z,l,h,r,R,s,1,2,k,K,v,t,D,f,I,O,B** are described jointly for 1d, 2d and 3d networks, in section 17.8.

Options **j,e,c,u,d,S,b,g,n,p** apply more specifically, or uniquely, to the 2d graphic, and are described below.

17.5.3 Moving or jumping between cells, 2d

To move the active cell, use the arrow keys. Alternatively, enter **j** to jump to a specific cell. The 2d cell position is specified according to the i,j coordinates. The following prompts are presented,

jump to index ij
enter i(35-0): enter j(35-0): *(this example for a 2d network size* 36×36*)*

17.5.4 Defining a block, 2d

Enter **b** in section 17.5.2 to define a 2d block of cells. The following top right prompt is presented,

2d block-2, all-a single cell-def: *(then, if* **2** *is selected...)*
2d block (this=26,24 max 39,39) *(for a* 40×40*)*
low corner (def 0,24) i: j:
high corner (def 0,24) i: j:

Enter **2** to define the block, **a** to define the whole network as a block. If **2** was entered, enter the low and high cell indices defining the block. The wiring of all the cells in the block will be shown in the graphic as direct links, colored light red. The active cell will still be visible and can be moved as usual.

For a 2d network shown in 1d, the prompt is as shown in section 17.4.4

An active block is indicated in the wiring graphic prompts (17.5.2), and the following options will apply to the whole block, for example,

...
rewire 2d block 15,15-25,25-30: hand-h rand-r special-s local:1d-1 2d-2
change k=8(max 13)-k rule:rev/trans-v/t
...

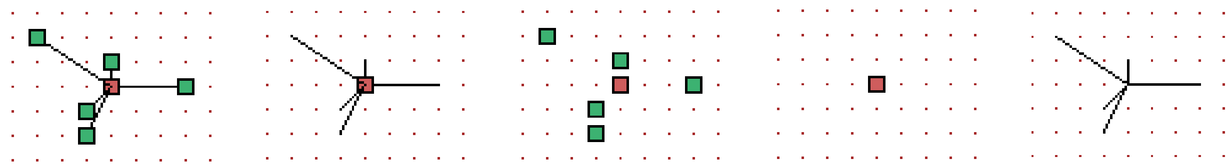
The rule options actually apply to the active cell, but the rule can then be copied to the bloc.

To deactivate the block enter **b** in section 17.4.2, then **return**.

17.5.5 Toggling the block, 2d

If a blog is active (see 17.5.4 above), enter **g** in section 17.5.2 to toggle the block, making it either visible or invisible, without deactivating it.

17.5.6 Alternative wiring presentation, 2d



1. cell+links+inputs 2. cell+links 3. cell+inputs 4. cell only 5. links only

Figure 17.12: Toggling between alternative ways of showing a cell and its connections in 2d.

A 5-way toggle allows alternative presentations of wiring, and the cells wired to, the active cell or block, which is especially useful in visualizing blocks.

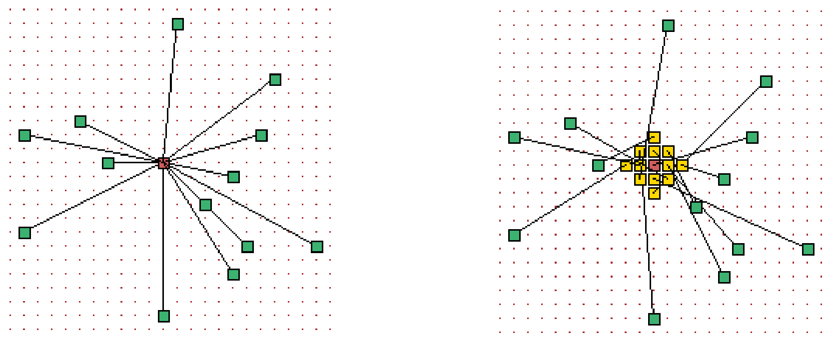
Enter **n** in section 17.5.2 to toggle between the five alternatives, as shown in figure 17.12. For a 2d block, the first four alternatives are illustrated in figure 17.14. On top of this, direct wiring or the pseudo-neighborhood can also be toggled as described in section 17.5.7 below.

17.5.7 Include the pseudo-neighborhood, or direct wiring only, 2d

The default wiring representation in 2d does not include the pseudo-neighborhood, but shows just the direct wiring as in figure 17.13a. Alternatively, the pseudo-neighborhood may be displayed as in figure 17.13b. Enter **p** to toggle between the two forms of representing the wiring.

17.5.8 Expand/Contract the scale, 2d

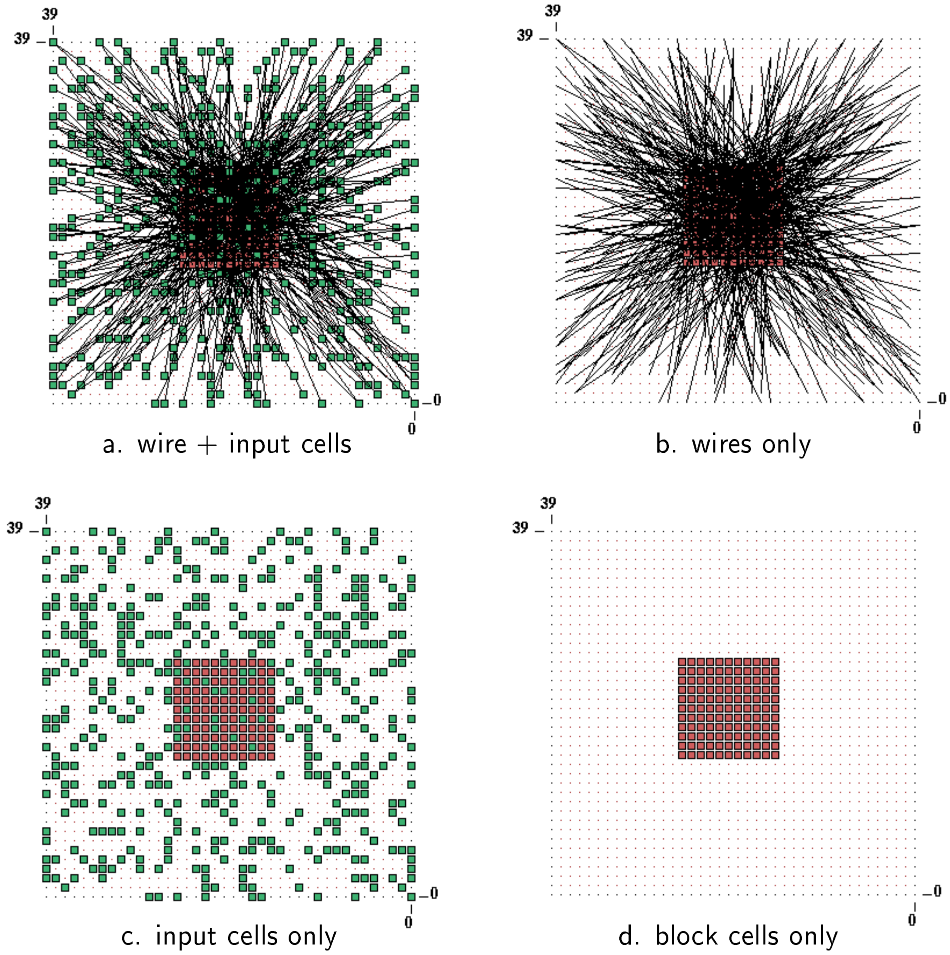
The scale of the 2d wiring graphic is set automatically to fit within its window. Enter **e** to expand, **c** to contract this default scale. The minimum contraction is one screen pixel for each network cell.



a. direct

b. pseudo-n'hood

Figure 17.13: Toggling between 2d wiring representations, k=13. (a) just direct wiring, or (b) including the pseudo-neighborhood.



a. wire + input cells

b. wires only

c. input cells only

d. block cells only

Figure 17.14: Toggling between 4 alternative presentations of a 2d block, defined at position 15,15 - 25,25 in a 40×40 network, $k = 1 - 9$, max-k=13, random wiring, as shown in figure 17.4.10.

17.5.9 Shifting the 2d graphic up and down

If only part of the graphic fits within the display area, it can be shifted up and down to see the relevant part. This is necessary for larger 2d networks, especially if they have been expanded in section 17.5.8.

Enter **u** or **d** to move the graphic up or down by 1/5 of its vertical dimension. Enter **s** to restore the default start position.

17.6 Wiring graphic, 3d

Enter **3** in section 17.1, for the 2d+3d graphic, where the network is shown simultaneously in 2d and 3d (see figure 17.15). As in 2d, this shows how a particular cell (the "active cell"), or a pre-defined 3d block of cells, is wired in the 3d network. The 2d and 3d graphics can be independently expanded and contracted.

The 2d graphic shows successive horizontal slices (levels), stacked above each other, with the levels indicated. For larger networks the 2d graphic can be shifted up and down to see the relevant part. Otherwise the 2d presentation is as described in section 17.5. The 3d version shows an axonometric projection seen from below, as if looking up at the inside of a cage. The active cell is moved around one level with the arrow keys, the square bracket keys [and] move up and down between levels, and its possible to "jump" to a new location. The rule for the active cell appears in the rule window described in section 16.14. As in the 2d wiring graphic (section 17.5), the display can be toggled between 'direct wiring" and wiring to the pseudo-neighborhood (as in figure 17.15). By default, "direct wiring" is displayed if the wiring is non-local, and to the pseudo-neighborhood for local CA type wiring. There is also a 4-way toggle to alter the presentation of connections, and the cells connected, which is especially useful in visualizing blocks. The current cell is colored red, its pseudo-neighborhood yellow, the "actual neighborhood" cells are colored green (see examples in figure 11.4.

17.6.1 Data - 3d wiring graphic

As for the 1d and 2d wiring graphics, various data about the current cell's wiring are shown at the foot of the 3d wiring graphic. For example, for figure 17.4.10 the data is as follows,

```
3d maxk=13 cell=9,5,7 k=3 wiring=8,9,4-14,4,10-9,7,7 outwires=2 k3-rule=(hex)20
```

See section 17.7 to decode this data. The current cell position, and the actual neighborhood positions, are shown in the form i, j, h. The rule and rule data are shown in the rule window (section 16.14).

17.6.2 Wiring graphic prompts, 3d

The following top right prompts/reminders appear at the same time as the 3d graphic. Various context sensitive options are presented for amending the wiring or rule/s for a single cell, a 3d block of cells, or for the network as a whole.

The presentation options allow for expanding and contracting the scale of the 2d and 3d graphic independently, and if only part of the 2d graphic fits within the display area, to move the network up and down to see the relevant parts.

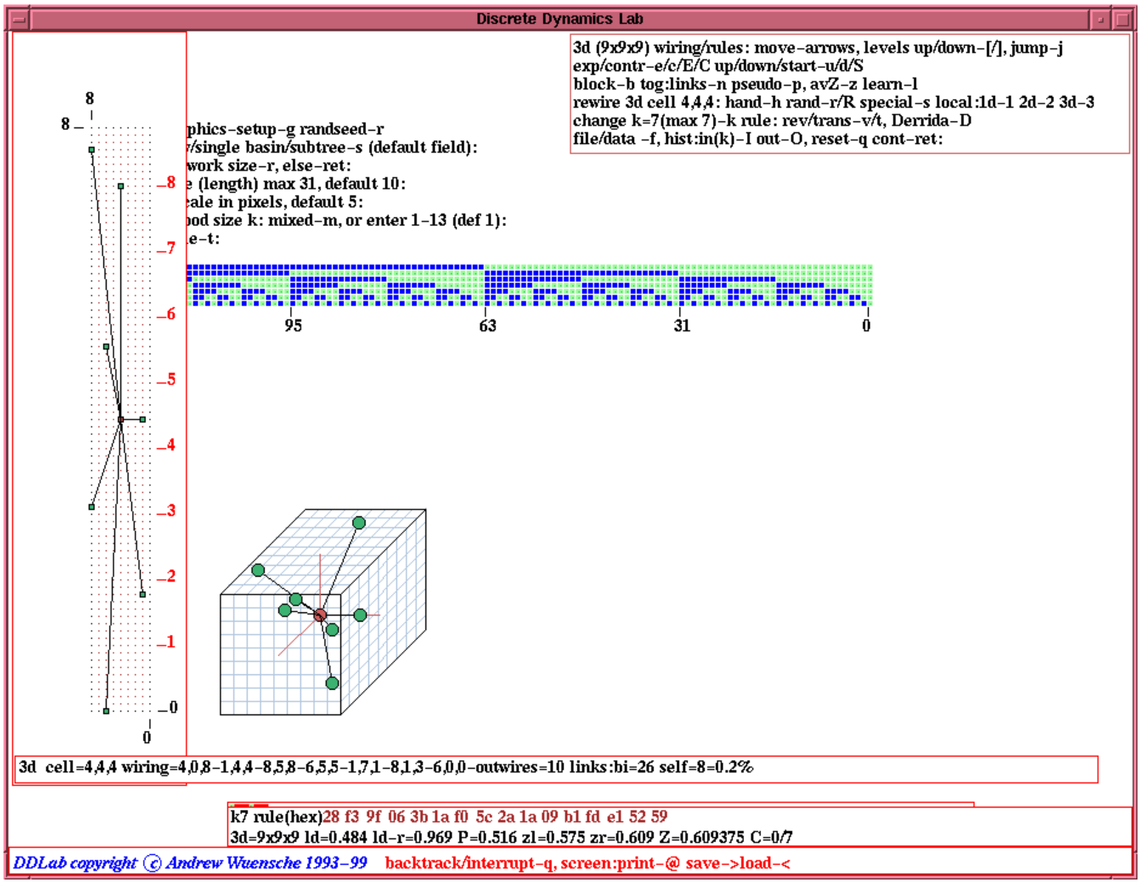


Figure 17.15: A 3d wiring graphic as it appears on the DDLab screen after the rules have been set. $n = 9 \times 9 \times 9$. k=7. *Left*: the network in 2d showing successive levels 0 to 9 stacked above each other. *Right*: the network in 3d, in an axonometric projection seen from below, as if looking up at the inside of a cage.

> **3d (20x20x20) wiring/rule: move-arrows, levels up/down-[/], jump-j**
> **exp/contr-e/c/E/C up/down/start-u/d/D**
> **block-b tog:block-g links-n pseudo-p, avZ-z learn-l** *(-g if block active)*
> **rewire 3d cell 5,5,5: hand-h rand-r/R special-s local:1d-1 2d-2 3d-3**
> **change k=6(max 13)-k, kill-K, rule:rev/trans-v/t, Derrida-D**
> **file/data-f hist:In(k)/Out/Both-I/O/B, reset-q cont-ret:**
> *(Note:* **-z** *for mixed rules,* **-l -v/t -D -f** *if rules set,* **-k** *if rules not set or mixed rules,*
> **-K** *if rules set and mixed-k)*

If the rules have not been set, the first line of the prompt reads,

> ... **- wiring only** *(instead of* ..., **wiring/rules***)*

Options **z,l,h,r,R,s,1,2,3,k,K,v,t,D,f,I,O** are described jointly for 1d, 2d and 3d networks, in section 17.8.

Options **j,e,c,E,C,u,d,S,b,g,n,p** apply more specifically, or uniquely, to the 3d graphic, and are described below.

17.6.3 Moving or jumping between cells, 3d

To move between network cells, use the arrow keys to move around one level. The square bracket keys [and] move up and down between levels, Alternatively, enter **j** to jump to a specific cell. The 3d cell position is specified according to the $i, j.h$ coordinates. The following prompts are presented,

> **jump to index ijh** *(this example for a 3d network size* $9 \times 9 \times 9$*)*
> **enter i(14-0):** **enter j(14-0):** **enter h(14-0):**

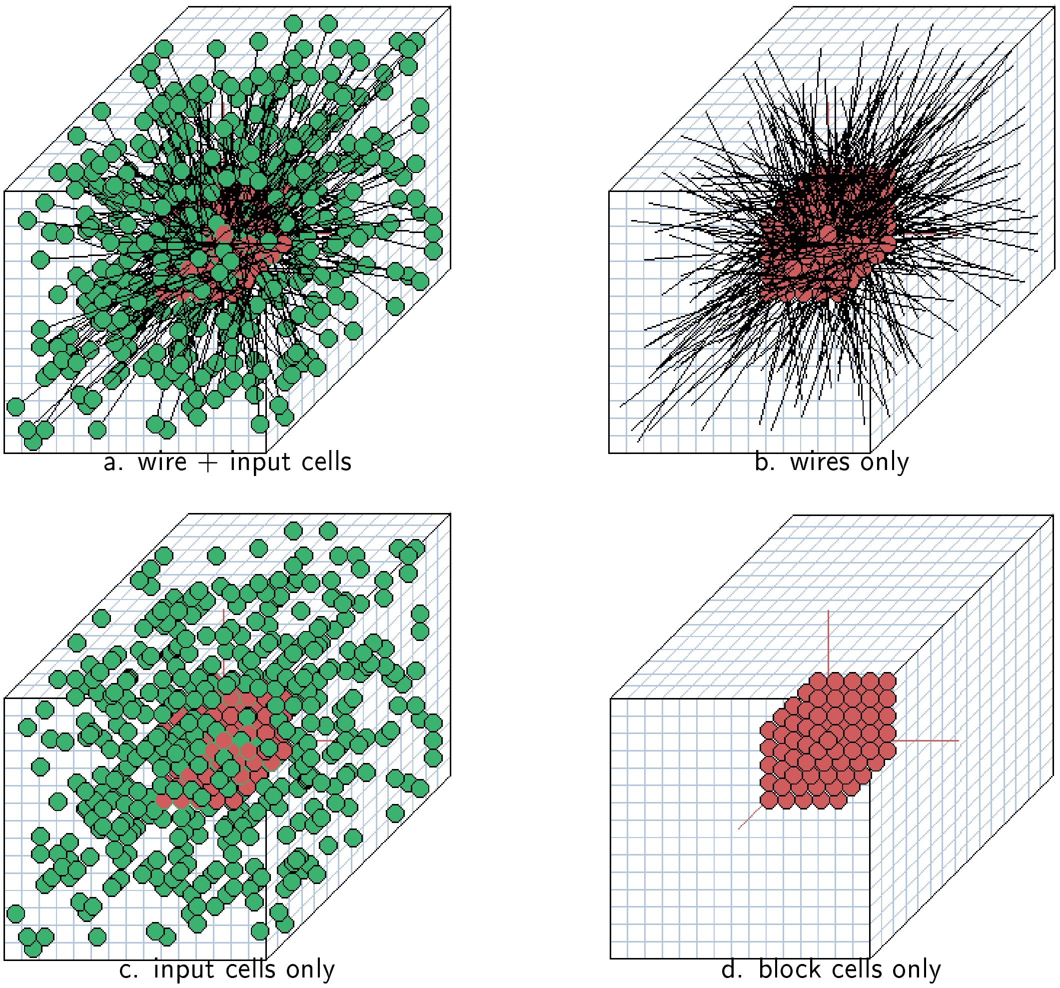


Figure 17.16: Toggling between 4 alternative presentations of a 3d block, defined at position 5,5,5 - 9,9,9 in a $15 \times 15 \times 15$ network, $k = 1 - 9$, max-k=13, random wiring.

17.6.4 Defining a block, 3d

Enter **b** in section 17.6.2 to define a 3d block of cells. The following top right prompt is presented,

3d range-3, all-a single cell-def: *(then, if* **3** *is selected...)*
3d block (this=7,7,7 max 14,14,14) *(for a* 40×40*)*
low corner (def 0,0,7) i: j: h:
high corner (def 0,24) i: j: h:

Enter **3** to define the block, **a** to define the whole network as a block. If **3** was entered, enter the low and high cell indices defining the block. The wiring of all the cells in the block will be shown in the graphic as direct links, colored light red. The active cell will still be visible and can be moved as usual.

For a 3d network shown in 1d, the prompt is as shown in section 17.4.4

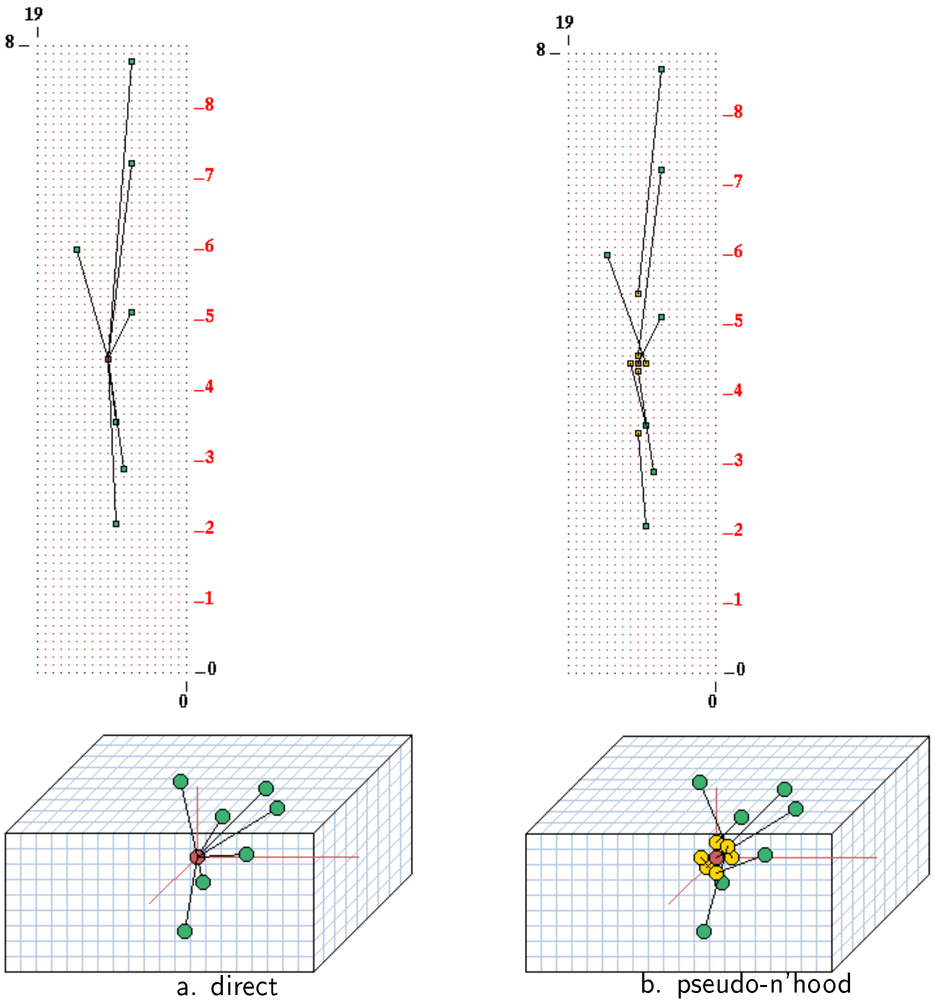


Figure 17.17: Direct (a) and pseudo-neighborhood wiring (b), shown as 2d representations (above), and 3d representations (below), of a 3d network ($20 \times 9 \times 9$), k=7.

An active block is indicated in the wiring graphic prompts (17.5.2), and the following options will apply to the whole block, for example,

...
rewire 3d block 9,9,9-15,15,15: hand-h rand-r special-s local:1d-1 2d-2 3d-3
change k=6(max 13)-k rule:rev/trans-v/t
...

The rule options actually apply to the active cell, but the rule can then be copied to the bloc. To deactivate the block enter **b** in section 17.4.2, then **return**.

17.6.5 Toggling the block, 3d

If a blog is active (see 17.6.4 above), enter **g** in section 17.6.2 to toggle the block, making it either visible or invisible, without deactivating it.

17.6.6 Alternative wiring presentation, 3d

A 5-way toggle allows alternative presentations of wiring, and the cells wired to, the active cell or block, which is especially useful in visualizing blocks.

Enter **n** in section 17.6.2 to toggle between the five alternatives, as shown in figure 17.18 For a 3d block, the first four alternatives are illustrated in figure 17.16.

On top of this, direct wiring or the pseudo-neighborhood can also be toggled as described in section 17.6.7.

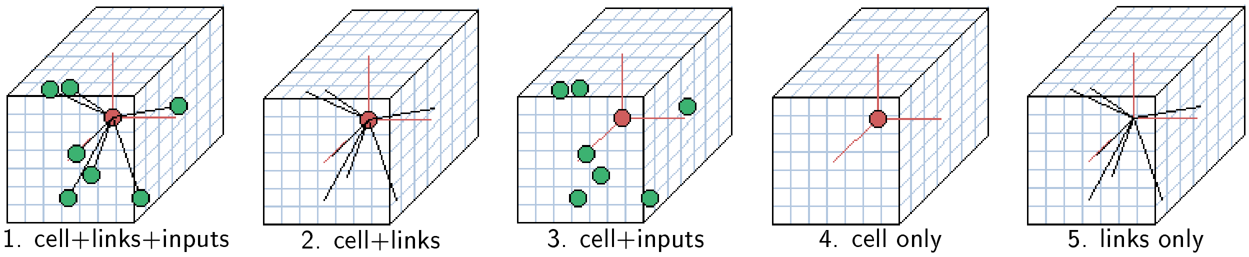


Figure 17.18: Toggling between alternative ways of showing a cell and its connections in 3d.

17.6.7 Include the pseudo-neighborhood, or direct wiring only, 3d

As in 2d, the default wiring representation in 3d does not include the pseudo-neighborhood, but shows just the direct wiring as in figures 17.17(a). Alternatively, the pseudo-neighborhood may be displayed as in figures 17.17(b). Enter **p** to toggle between the two. On top of this, further alternative wiring presentations can be displayed as described in section 17.6.6.

17.6.8 Expand/Contract the scale, 3d

The scale of the 2d and 3d versions of the 3d wiring graphic can be independently expanded and contracted. Enter **e** to expand, **c** to contract the 2d version. Enter **E** to expand, **C** to contract the 3d version. The minimum contraction in both cases is one screen pixel for each network cell.

17.6.9 Shifting the 3d graphic up and down

If only part of the 2d version of the 3d wiring graphic fits within the display area, it can be shifted up and down to see the relevant part. This is necessary for larger 3d networks, especially if they have been expanded in section 17.6.8.

Enter **u** or **d** to move the graphic up or down by about one level. Enter **s** to restore the default start position.

17.7 Decoding wiring graphic data - 1d, 2d and 3d

Various data about the active cell's wiring are shown at the foot of the 1d, 2d and 3d wiring graphic. For example,

for 1d

cell=14 wiring=24 15 4 1 7 outwires=3 bi-links=7 k5-rule=(hex)8a f4 be f4

for 2d

2d maxk=13 cell=20,20 k=9 wiring=1,5 39,6 13,38 22,2 12,21 25,36 20,37 8,30 17,6 outwires=3 bi-links=15

for 3d

3d maxk=13 cell=9,5,7 k=3 wiring=8,9,4-14,4,10-9,7,7 outwires=2 k3-rule=(hex)20

decode of wiring data

maxk– For mixed k networks. Shows the network max-k, which may be greater than the actual maximum k.

cell –The cell index, in 1d, 2d or 3d.

wiring–Network indices wired to the pseudo-neighborhood, ordered $k-1\ldots 0$, in 1d, 2d or 3d..

outwires–The number of output wires, how many wires from the network are "plugged into" to the cell in question.

bi-links –The number of cell pairs that have both inputs and outputs to each other.

k5-rule –The rule in hex if $k \leq 5$.

17.8 Further options for the 1d, 2d and 3d wiring graphics

The following options in sections 17.4, 17.5 or 17.6 apply jointly to 1d, 2d and 3d networks, and are described in sections 17.8.1 - 17.8.12 below,

avZ-z ...	compute the (weighted) average λ and Z, section 17.8.1 for a mixed rule network.
learn-l ...	learning pre-images without attractor basins, section 17.8.2.
hand-h ...	hand rewiring, section 17.8.3.
rand-r/R ...	random rewiring, section 17.8.4.
rand-r/R ...	randomly rewire with a bias according k in mixed-k networks, section 17.8.4.
special-s ...	special wiring, biase the random rewiring, section 17.8.5.

local:1d-1 2d-2 3d-3 ... set local CA wiring (1d, 2d or 3d) for the cell or block, section 17.8.6.

change k=8(max 13)-k ... change the neighborhood size, k, for the cell or block, section 17.8.7.

kill-K ... kill or neutralize a cell by cutting all its links, both inputs and outputs, except for one input to itself, section 17.8.8. The k=1 rule is set to 2, meaning its value stays constant, with effective k=0.

rule:rev-v ... revise the rule for a cell and copy to a block, section 17.8.9.

rule:trans-t ... transform the rule or rules, set Canalizing inputs, section 17.8.10.

Derrida plot-D ... draw the Derrida plot for the network., section 17.8.11.

file/data-f ... filing, save and load network architecture, section 17.8.12.

hist:In(k)/O/B-I/O/B ... show a histogram of the frequency distribution of inputs (i.e. k), outputs, or both (i.e all connections) in the network.

17.8.1 Computing the (weighted) average λ and Z parameters

Applies once a rule-mix has been set in chapter 13, with or without random wiring

Enter **z** in sections 17.4, 17.5 or 17.6, to calculate and display the average, and weighted average, λ_r and Z parameters of the rules making up the rule-mix.

This data is displayed in a top right window, for example, for a 1d network $n = 150$, $k = 3 - 7$, and randomly assigned rules,

av:ld-r=0.839063 Z=0.63653
wt.av:ld-r=0.826438 Z=0.626692 cont-ret:

For random wiring and/or mixed k networks, the weighted averages take account of the influence that each cell has on the network according to is proportion of the network's out-wires, represented graphically by the height of cells at t_0 in a 1d wiring graphic, for example in figure 17.4.1. In networks with regular wiring and no rule mix, the weighted average equals the average.

17.8.2 Learning pre-images without attractor basins

Requires random wiring or a rule-mix (set in section 13), preferably both

Enter **l** in sections 17.4, 17.5 or 17.6 to start the "learning/forgetting" routines for attaching/detaching sets of states as pre-images of a target state. The network's wiring and/or rule scheme is automatically amended to achieve the required transitions between states. The learning routine is described in detail in chapter 34, where learning is also invoked in the context of "sculpting attractor basins". In this case, the results and side affect of learning are displayed when attractor basins are generated, but this places a limit on the size of the network (see section 7.4). Invoking the routine at this point in the program, or when running networks "forwards only", allows much larger networks to "learn".

Note that the full scope of the learning routine requires a network with random wiring and mixed rules. If the network has regular 1d wiring, learning by rewiring is not applicable. If the network is set up with a single rule, learning by flipping bits in rule-tables is not applicable. Note that regular 1d wiring can be treated as random (see section 11.4.1), and a rule mix where all the rule are the same can be set up as described in section 12.6.

17.8.3 Hand rewiring

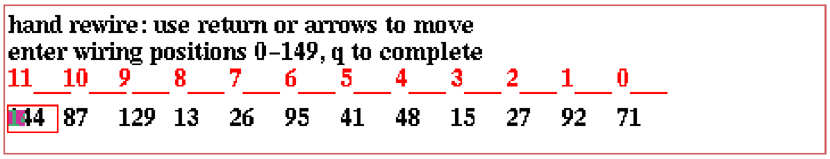


Figure 17.19: Hand wiring a single cell from the 1d wiring graphic, n=150, k=13

Enter **h** in section 17.4.2, 17.5.2 or 17.6.2, to rewire individual wires for just the active cell by hand. The block set in section 17.4.4, 17.5.4 or 17.6.4 does not apply in this case.

The cell is rewired in a way similar to that described in section 11.6. A top right window displays the pseudo-neighborhood wiring connections as a 1d "spread sheet" with k entries, (figure 17.19), including the following reminder, for example,

hand rewire: use return or arrows to move
enter wiring position 0-149, q to complete *(this example for a 1d network, size 150)*

Note that the cell index appears as if for 1d, even if the network is 2d or 3d.

17.8.4 Random rewiring

Enter **r** or **R** in section 17.4.2, 17.5.2 or 17.6.2, to randomly reset the wiring of the active cell, or of the block if defined (in section 17.4.4, 17.5.4 or 17.6.4), and visible.

If **r** is entered, the biases on random wiring set in section 11.5, and 17.8.5 below, will be respected. If they are not set there will be no bias.

If **R** is entered, the wiring will be randomly reset, but biased by k for each cell in a mixed k network. That is, the probabitity of wiring to a cell with k inputs will be given by $P = k/T$, where T is the total number of links in the network.

If a power-law distribution of k was set for the network in section 8.8.2, this also gives an approximate power-law distribution of outputs, creating a "scale free" network. This is supposed to be characteristic of many natural and artificial networks, from the internet to metabolic networks [1]. The graph of such a network is shown in figure 20.

17.8.5 Biased random rewiring

If **s** (for special) is selected in section 17.4.2, 17.5.2 or 17.6.2, a series of prompts are presented in a top right window that allow a variety of restrictions or biases to the random wiring before it is set in section 17.8.4 above. These biases apply to the active cell, or to the block if defined (in section 17.4.4, 17.5.4 or 17.6.4), and visible.

A series of prompts are presented in a top right window. The options differ for 1d, 2d or 3d wiring graphics. They are same the as in *Special wiring, random*, section 11.5, where the options and prompts are described.

17.8.6 Regular 1d, 2d or 3d wiring

If **1**, **2** or **3** is selected in sections 17.4.2, 17.5.2 or 17.6.2, the wiring of the active cell, or block if defined (in section 17.4.4, 17.5.4 or 17.6.4), and visible, will be reset to regular 1d, 2d or 3d CA-like wiring to the local neighbourhood, with periodic boundary conditions.

For 3d networks, 1d, 2d or 3d regular CA wiring applies. For 2d networks, 1d, 2d regular wiring CA applies. For 1d networks, only 1d regular CA wiring applies.

Note that CA wiring can also be set as "random wiring" in section 17.8.5 above, allowing various biases to the CA wiring as described in section 11.5.

17.8.7 Changing the neighborhood size, k

Wether or not the network has a neighborhood mix, **k** may be selected in sections 17.4.2, 17.5.2 or 17.6.2, to change the neighborhood size k of the active cell, or of the block if defined (in section 17.4.4, 17.5.4 or 17.6.4), and visible. The following additional prompt appears,

> **reset nhood size k (1-13):** *(if max-k=13)*

If a block is visable in the graphic, the change of k will apply to all cells in the block.

The maximum k allowed is *max-k*, whick may be higher than the actual k in the network (see section 8.11), with an upper limit of k=13. The new wiring will preserve as much as possible of the old neighborhood wiring. If k is increased, the wiring at the extra pseudo-neighborhood indices is set as regular 1d, 2d or 3d (depending on the network) with periodic boundary conditions.

For a honogenious k network, the individual cells may only have their k setting reduced. The network will from then on be treated as a mixed-k network.

If a rule scheme has been set (see section 12), the existing rule table (or part of it if k was reduced) will be preserved. Any excess rule-table entries are set to 0.

17.8.8 Kill a cell

Applies only to mixed-k networks, and after the rule scheme has been set in chapters 12 - 16

Enter **K** in sections 17.4, 17.5 or 17.6, to kill or neutralize a cell by cutting all its links, both inputs and outputs, except for one input to itself, thus reducing k to one. The k=1 rule is set to 2, meaning its value stays constant, with effective k=0 (see 8.2), so the cell has no influence on the network. For 1d networks, a cell can also be deleted entirely (see section 17.4.10).

17.8.9 Revising and copying the rule

Applies only after the rule scheme has been set in chapters 12 - 16

Enter **r** in sections 17.4.2, 17.5.2 or 17.6.2, to revise the rule of the active cell. Once the rule is revised, it can be copied to a block.

This option is intended for networks with mixed rules. For a single rule network (set in section 12.5), i.e. without a rule-mix, any revision applies to the whole network. Note that a rule-mix where all the rule are the same can be set up as described in section 12.6.

A secondary window is presented in the lower right hand corner of the screen, with rule selection prompts. The various methods for revising and re-selecting rules are the same as in *Setting a singe rule* described in chapter 16.

If a block was defined in sections 17.4.4, 17.5.4 or 17.6.4, and if the block is visible, the rule can be automatically copied to all cells in the block which have the same k as the cell in question. The following prompt is presented in a top right window,

for a 1d wiring graphic
Copy rule to 33-55 -c:
for a 2d wiring graphic, 40×40
Copy rule to 2d range 0,22-39,33 (880-1359)-c:
for a 3d wiring graphic, $20 \times 20 \times 20$
Copy rule to 2d range 0,0,8-19.19.12 (3200-5159)-c: *(values shown are examples)*

Enter **c** to copy the rule within the block, to all cells with matching k.

Once set (and copied), the new rule is displayed in the rule window (see section 16.14).

To change a rule to one with a different neighborhood size k, first change k in the rewiring window (see section 17.8.7), then change the rule.

17.8.10 Transforming the rule

Applies only after the rule scheme has been set in chapters 12 - 16

If **t** is selected in sections 17.4.2, 17.5.2 or 17.6.2, the rule for just the active cell may be transformed in the various ways described in detail *Transforming network rules*, chapter 18.

This option is intended for networks with mixed rules. For a single rule network (set in section 12.5), i.e. without a rule-mix, any revision applies to the whole network.

A top right window is presented with the transformation prompts. The rule may be transformed to equivalent or related rules. For example the rule may be complimented, or transformed to an equivalent rule by negative or reflection transformations. Neutral tranformations may be made to rules with greater k, or k reduced to "effective k" for the particular cell or for the whole network. Canalizing inputs may be set or amended for the active cell or the whole network as described in section 15.

The network may be "reverse engineered" by loading an exhaustive mapping of transitions (see section 18.12) and automatically generating the minimal mixed-k network that satisfies the mapping, (i.e. reduced to effective k), one solution to the "inverse problem".

17.8.11 Derrida plot, from the wiring graphic

Enter **D** in sections 17.4.2, 17.5.2 or 17.6.2, to generate the Derrida plot for the network, described in chapter 22.

17.8.12 Filing, from the wiring graphic

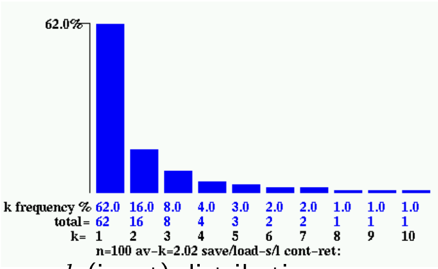
The filing option does not appear and is not active until rule/s have been set

Once the rule or rules have been set, enter **f** in sections 17.4.2, 17.5.2 or 17.6.2, for the network filing options described in chapter 19, *Save/load/print network architecture*. To load a network file, first set up a similar "dummy" network.

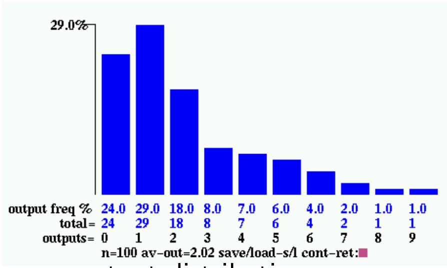
The whole network or just the wiring or rule-mix can be saved or loaded to a file. This includes the k-mix if any. If a network file smaller than the base network is loaded, it will automatically be defined as a block in 1d, 2d or 3d (see sections 17.4.4, 17.5.4 or 17.6.4), and shown in the base network as in figures 17.4.1, 17.14 and 17.16. Any number of sub-networks can be loaded into the base network.

A network description can also be printed to a file, to the xterm window for UNIX and Linux, or to a printer for DOS. For further details see chapter 19, *Save/load/print network architecture*.

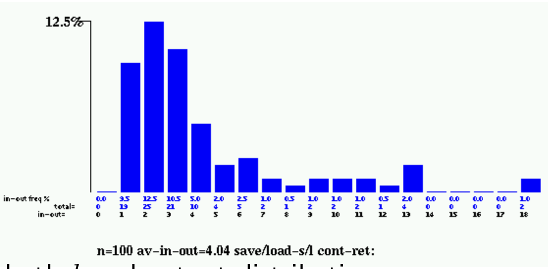
17.8.13 The histogram of the network's k and output distribution



k (input) distribution



output distribution



both k and output distribution

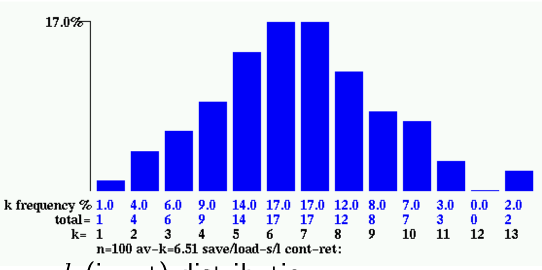
Figure 17.20: A histogram of *above left*: the k (i.e. input) distribution in the network, and *above right*: the output distribution. Both have been set to follow a power-law. *left*: the distrbution of k and output combined. The power-law exponent was set as 2.0 in section 8.8.2. Random wiring was then biased by k in section 17.8.4.

Enter **I**, **O** or **B** 17.4.2, 17.5.2 or 17.6.2, to show a histogram of the frequency distribution of node connections. **I** gives the k (i.e. input) distribution, **O** gives the output distribution, and **B** gives both the input and output, i.e of all connections in the network.in the network. The k distribution can also be displayed from section 8.8.3.

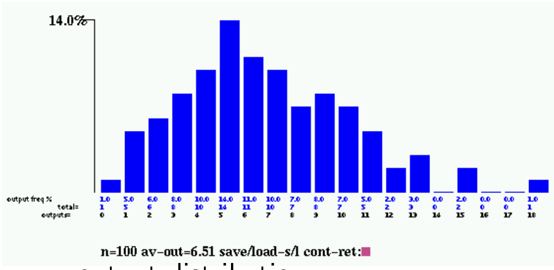
Examples of power-law and normal (Poisson) distributions are shown in figures 17.8.13 and 17.8.13.

The histogram plots each k or output (x axis), against its frequency in the network as a percentage (y axis). The actual frequency, as a percentage and total, are shown under each histogram bar.

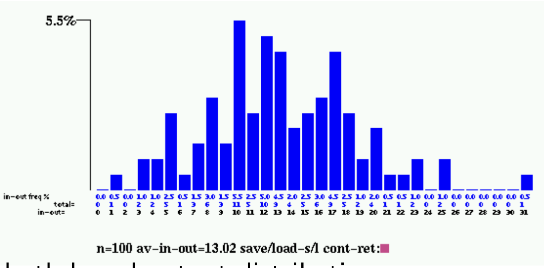
The following information and prompt is also shown,



k (input) distribution



output distribution



both k and output distribution

Figure 17.21: A histogram of *above left*: the k (i.e. input) distribution in the network, and *above right*: the output distribution. Both have been set to follow a nornal distribution. *left*: the distrbution of k and output combined. The average k was set to 6.5 in section 8.7. Random wiring was then biased by k in section 17.8.4.

n=100 av-k=2.02 save/load-s/l cont-ret: *(for example)*

n is the network size which should equal the sum of all the frequency totals. **av-k**, **av-out** or **av-in-out** is the average k, out-degree, or all links to a node.

Enter **s** or **l** save or load the histogram data as a `.his` file (see Filing, chapter 35). In both cases the data appear in the xterm window (*not for DOS*), for example for figure 17.8.13(*left*),

```
histogram: 1+13 columns
0 635 159 71 40 26 18 13 10 8 6 5 5 4
```

Chapter 18

Transforming network rules

Rule transformation options are presented after a single rule has been set (see section 16.16), and also from the wiring graphic prompts (see 17.8.10. For a rule mix, rule transformations from the wiring graphic may be applied to the active cell, or to the whole network.

Various transformations may be made to produce equivalent or related rules. For example rules may be complimented, or transformed to an equivalent rule by negative or reflection transformations, within their equivalence classes and rule clusters [12].

The rule may be inverted, so that a rule-table set out according to a convention where the most significant bit is 0 may be transformed into the equivalent rule-table where the most significant bit is 1, the convention in DDLab. Neutral transformations may be applied to produce rules with greater k, or to rules with smaller "effective k".

If a mixed k network was set up where k_{max}=n, an option is presented to load an exhaustive mapping of transitions (see Filing, chapter 35) and "reverse engineer" the network. An algorithm automatically transforms the network to satisfy the mapping. The "effective k" transformation may then be applied to reduce the network back to its minimal k-mix. A further option allows max-k, which may be greater than the maximum k found in the network, k_a, (see section 8.11), to be reduced to k_a.

The canalizing inputs of the rule may be set or amended. The (changed) rule may be saved as a `*.rul` file (see chapter 19).

Note that transforming the rule of an individual network cell in isolation only applies to a mixed rule network (see 12.5. However, a rule-mix where all the rules are the same can be set up as described in section 12.6. For a single rule network without a rule-mix, any transformation applies equally to all cells in the network.

18.1 Options for transforming rules

The options for transforming rules appear in a top right window after single rules have been set (see section 16.16), or from the wiring graphic (see section 17.8.10). The options depend on the type of network, and transformations may be made cumulatively. The altered rule is displayed in the lower rule window. Examples are show below,

18.1.1 Transform options, single rule

For a single rule network, transformations apply to all cells in the network.

transform rule: solid-o invert-v comp-c neg-n ref-r canal-C
equiv$>$k (4-13), eff k: all-k, save-s: *(if k=3, values shown are examples)*

18.1.2 Transform options, mixed k

For a mixed k network, transformations may apply to one cell or all cells in the network, and there are further options to load an exhaustive map, a `.exh` file (see Filing, chapter 35), if $k_{max} = n$ and $n \leq 13$. The exhastive map would have been saved in section 29.5.3.

transform rule: solid-o invert-v comp-c neg-n ref-r canal-C (all+a)
equiv$>$k (4-7), max k-m, eff k: all-K this-k, save-s: *(if k=3 and k_{max}=7)*

if $k_{max} = n$ and $n \leq 13$, an exhustive map can be loaded
exh map-e, equiv$<$k (4-13), eff k: all-K this-k, save-s:

To transform all cells in the mixed rule network with options **o, v, c, n, r** enter the option followed by **a**. For example to compliment all rules enter **ca**. Similarly, to revise canalizing inputs for the whole network (see chapter 15, enter **Ca**.

18.1.3 Transform options, mixed rule, homogenious k

For a mixed rule network with homogenious k network, the "greater k" and "effective k" transformations do not apply.

transform rule: solid-o invert-v comp-c neg-n ref-r canal-C (all+a)
save-s:

Note that a mixed k network may be set up where all rules and k's are the same, allowing the full scope of the transformations.

18.2 Solidifying the rule

If **o** (or **oa**) is entered at the prompt in section 18.1, the rule (or rules) will be changed so that an "on" (i.e. "1") cells remains on. This is of interest mainly for CA, or for RBN where the central wire couples to itself.

18.3 Inverting the rule

If **v** (or **va**) is entered at prompt in section 18.1, the rule (or rules) will be inverted, transformed by swapping the outputs of complimentary neighborhood pairs in the rule-table. For example (for k=5) the outputs of 00000 and 11111, or 00111 and 11000, are swapped. In DDLab, the rule-table convention makes the most significant bit corresponds to the all 1's neighborhood (see section 12.1) which allows the rule-table bitstring to be expressed simply as a decimal number (as well as hex), whereas some other conventions[6] make the most significant bit the all 0's neighborhood. For example the k=7 "density" rule (in hex)

```
05040587 05000f77 03775583 7bffb77f
```

is transformed to

```
feedffde c1aaeec0 eef000a0 e1a020a0
```

18.4 Complimentary transformation

If **c** (or **ca**) is entered at the prompt in section 18.1, the rule (or rules) will transformed to its "compliment", by flipping all entries in the rule-table, 1 to 0 and 0 to 1.

18.5 Equivalent rule by the Negative transformation

If **n** (or **na**) is entered at the prompt in section 18.1, the rule will be transformed into its "negative" equivalent rule, (by swapping the outputs of complimentary neighborhood pairs in the rule-table, then transforming the resultant rule-table into its compliment. A negative rule and the start rule have equivalent dynamics, but with all state configurations in the space-time pattern complimented.

Note that a negative transformation followed by a reflection transformation (see section 18.6), or vice-versa (the order is equivalent) produces another equivalent rule, making 4 in all.For certain rules the transformations result in identity.

18.6 Equivalent rule by the Reflection transformation

If **r** (or **ra**) is entered at prompt in section 18.1, the rule (or rules) will be transformed into its equivalent rule by "reflection" (by swapping the outputs of pairs of asymmetric reflected neighborhoods in the rule-table). A reflected rule and the start rule have equivalent, but reflected dynamics.

Note that a reflection transformation followed by a negative transformation, (#18.5), or vice-versa (the order is equivalent) produces another equivalent rule, making 4 in all (for certain rules the transformations result in identity).

18.7 Setting canalizing inputs, single rule

If **Ca** is entered at the prompt in section 18.1 for a mixed rule network, canalizing inputs can be randomly biased for the whole network as described in chapter 15

If just **C** is entered, canalising inputs may be amended to for a single rule network, or for the active cell in a mixed rule network. A given number of the rule's imputs may be set to be canalizing, either explicitly or at random. The following top right prompt is presented.

set canalizing inputs: explicitly-e, number at random-n:

18.7.1 Canalizing inputs at random, single rule

Enter **n** to set a given number of inputs as canalizing at random (which can be reduced as well as increased). The following prompt appears, enter the required number,

canalizing inputs=0, set new number (0-5): *(values shown are examples)*

18.7.2 Canalizing inputs explicitly, single rule

Enter **e** to set the canalizing inputs explicitly, the following prompts appear in sequence,

wire (0-5): **input value (1,0):** **output value(1,0):** *(values shown are examples)*

Enter the neighborhood index (wire) to be set as canalizing, and the input and output canalizing values (0 or 1).

18.8 Transform all cells in the network

For a single rule network, transformations described in sections 18.2 to 18.7 apply to all cells, whereas for a mixed rule or mixed-k network, just the selected cell in the wiring graphic (see section 17.8.10) can be transformed, or all cells in the network. To transform all cells, add **a** to the selection, i.e. enter **oa** in section 18.2, **va** in section 18.3, **ca** in section 18.4, **na** in section 18.5, **ra** in section 18.6 and **Ca** in section18.7.

Entering **Ca** allows canalizing inputs for the whole network to be to be reset as described in 15.

18.9 Equivalent rules with greater k

A rule with a given neighborhood size k has equivalent rules with any greater value of k. If $k < 13$ (maximum k currenty supported by DDLab), or $k < k_{max}$ for a mixed k network, a greater value of k may be entered at the prompt in section 18, to produce a neutral transformation of the rule. The algorithm is designed for 1d networks; for 2d and 3d the transformation is not neutral.

For instance, a k=3 rule may be transformed to a k=5 or k=9 rule. Although the larger k rule's dynamics is identical, mutations of the larger k rule are finer grained, so the transformation is useful for looking at close mutations of a given rule.

For networks with a homogeneous rule the transformation applies to the whole network. In mixed k networks, only the chosen cell in the network is transformed. For mixed rule, homogenious k networks, the option does not apply.

An extra wire added to an even-k pseudo-neighborhood is added on the left, at the new index k-1. An extra wire added to even-k is added at index 0 (on the right), the original wires are shifted one place to the left. Thus the original pseudo-neighborhood remains central in the enlarged pseudo-neighborhood. The rules are transformed accordingly so that the added wires are redundant and play no role in network dynamics. However, if the rule is mutated the extra wires would come into play. The extra wires are connected within the network as they would be in a 1d cellular automata.

18.10 Reducing k_{max} to the maximum k in the network

For mixed k netorks, k_{max} may be larger than the maximum k in the network. This may have been deliberatly set in section 8.11, or a network may have been loaded with a smaller maximum k than the base network (see section 19.8.5).

Enter **m** to reduce max-k to the maximum k in the network.

18.11 Effective k

The pattern of 0s and 1s in a rule-table may be such that some wires are redundant, playing no role in the dynamics. An example is a rule that has been neutrally transformed with greater k, in section 18.9 above. If **k** or **K** is entered at the prompt in section 18 and redundant wires exist, the rule will be neutrally transformed to a rule with decreased k (to a minimum of k=1) by pruning redundant connections and transforming the reduced rule-table appropriately.

For a single rule network, the effective k transformation applies to the whole network. For mixed k networks, if lower-case **k** is entered only the chosen cell in the network is transformed, whereas if upper-case **K** is entered all cells will have their rules reduced to each effective k (to a minimum of k=1). To make the effective k 0, see section 13.2.

18.12 Reverse engineering - loading an exhaustive map

For a network were $k_{max} = n$ and $n \leq 13$, an exhaustive list of transitions, a random map, that was previously created (see section 29.6) may be loaded to "reverse engeener" the network by transforming it to satisfy the mapping. This is a rudimentary solution to the inverse problem. At present it is restricted to the case where all transitions are specifed, and where $k_{max} = n$, so network sizes $n \leq 13$ because k_{max}=13 in DDlab at present.

The resulting network's dynamics, and attractor basins, will correspond to the mapping. The method works in three stages, firstly the rules and wiring are transformed for a fully wired network where $k = n$ for all cells, then the effective k option (see section 18.11) is implimented, followed by the option, if required, to reduce k_{max} to the actual maximum k found in the network (see section 18.10.

1. Enter **e** at the prompt in section 18 to load the mapping file and transform the network to a fully wired network. If you wish, you may see the fully wired transformed network by pressing "return" and reverting to "Reviewing network architecture, wiring and rules" (section 17).

2. Enter **K** to reduce the wiring to effective k (see section 18.11). There will be a pause while this is being computed.

3. Enter **m** to reduce k_{max} to the maximum k found in the network (see section 18.10).

The result may been seen by reverting to "Reviewing network architecture, wiring and rules" (section 17), and by generating attractor basins (section 30) to check that they are the same as the basins for the network that was used to generate the exaustive map.

18.13 Saving the transformed rule

The transformed rule is displayed in the rule window. Enter "**s**" at prompt in section 18.1.1 to save the rule (see Filing, chapter 35).

Chapter 19

Save/load/print network architecture

not applicable to regular 1d CA, or if rule/s have not been set

Once the rule or rules have been set, enter **f** at the network architecture prompts in sections 17.4.2, 17.5.2 or 17.6.2, for the network filing options described in this chapter.

The network architecture data may be saved, loaded, or printed to a printer or file, or to the xterm window (UNIX and Linux). The k-mix, rule scheme and wiring scheme may be treated in isolation, or combined. Depending on the type of network, one of the prompts below (19.1- 19.3) is displayed in a top right window. If the save or load option, or printing to a file, is selected, top right filing windows appear to select the filename etc. (see Filing, chapter 35). The file types are described in section 19.6.

When loading a network, it is first necessary to set up an appropriate "dummy" network with similar attributes. The whole network or just the wiring or rule-mix can be saved or loaded to a file. If a network file smaller than the base network is loaded, it will automatically be defined as a block in 1d, 2d or 3d (see sections 17.4.4, 17.5.4 or 17.6.4), and shown in the base network as in figures 17.4.1, 17.14 and 17.16. Any number of sub-networks can be loaded into the base network.

19.1 Mixed rules, non-local wiring, mixed-k

For a network with mixed rules and non-local wiring the following prompt is presented,

for mixed k networks
save/load/print mixed k network: save mix-k
just wiring: print-pw, save-sw
just rules: print-pr, save-sr
rulemix+wiring: print-pb, load-lb, save-sb:

for homogenious k networks
save/load/print uniform k network:
just wiring: print-pw, load-lw, save-sw
just rules: print-pr, load-lr, save-sr
rulemix+wiring: print-pb, load-lb, save-sb:

A file with just a wiring scheme, or just a rule scheme, can only be loaded into a homogenious k network. Only a combined rulemix+wiring file can be loaded into a mixed k network (see 19.8). The k-mix information is held as part of the wiring scheme and so is loaded when loading "rulemix+wiring". Just the k-mix may be saved by entering **k**, but the k-mix *in isoloation* may only be loaded earlier in the program sequence (see sectin 8.5).

Subject to these constraints, to save, load or print the network architecture enter two key strokes as follows,

firstly	*followed by*
p - to print	**w** - for wiring
l - to load	**r** - for rules
s - to save	**b** - for both wiring and rules

For example, enter **sw** to save a wiring scheme only file (`.wso`), **sr** for a rule scheme only file (`.rso`), or enter **sb** to save a file that combines both wiring and rule scheme, and possibly the k-mix, (`.wrs`).

19.2 One rule and non-local wiring

For a network with just one rule but non-local 1d wiring the following prompt is presented,

wiring: print-p, load-l, save-s:

Enter **p**, **l** or **s** to print, save or load the wiring scheme.

19.3 Mixed rules and local 1d wiring

For a network with mixed rules but local 1d wiring the following prompt is presented,

rulemix: print-p, load-l, save-s:

Enter **p**, **l** or **s** to print, save or load the wiring scheme.

19.4 Saving just the k-mix

Enter **k** in section 19.1 to save just the k-mix (see also section 8.12.3). A filing prompt box will allow saving a `.mix` file (the default filename is `mymix.mix` (see Filing, chapter 35). Loading the k-mix in isolation from a `.mix` file is only possible towards the start of DDLab as described in section 8.5. However, information on the k-mix in mixed k networks is contained in the `.w_s`, `.r_s` and `.wrs` files (see section 19.6), so does not need to be saved separately.

19.5 Printing network data to the xterm/printer or file

If **p** is selected in sections 19.1- 19.3 above, the following prompt is presented,

> *for DOS*
> **network text data: to printer-p, file-f both-b:**
> *for UNIX or Linux*
> **network text data: to xterm-p, file-f both-b:**

In DOS enter **p** to print the data to a printer, **f** to save the ascii data to a `.dat` file, or **s** to do both simultaneously. In UNIX or Linux enter **p** to print the data to the xterm window, **f** to save the ascii data to a `.dat` file, or **s** to do both simultaneously.

Before printing to a file, top right windows will prompt for the file name (default `net_data.dat`, (see Filing, chapter 35). The format of the network text (i.e. ascii) data, and the decoding of the data, is shown in the examples in below.

19.5.1 Printing network data - homogenious k

Example of printing network data (to a printer for DOS, to an xterm for UNIX, or to a file) for a network with mixed rules and non-local wiring, but homogeneous neighborhood size.

```
data code
k=3, n=10 neighborhood size k, network size n
cell.  wiring(2-0) rule(hex) ld ld-r Z C reminder of data order (see below)
-------------------------------------
9.  7 3 1, ec 0.625 0.75 0.625 1/3 *1* cell index, wiring, rule in hex, rule parameters
8.  0 7 2, ee 0.75 0.5 0.5 2/3 *10
7.  8 9 5, 5a 0.5 1 1 0/3
6.  7 8 0, ee 0.75 0.5 0.5 2/3 *10
5.  0 6 2, 74 0.5 1 0.5 0/3
4.  5 9 1, e8 0.5 1 0.5 0/3
3.  6 3 1, 9a 0.5 1 1 0/3
2.  8 4 6, 46 0.375 0.75 0.75 0/3
1.  3 7 4, 4c 0.375 0.75 0.625 1/3 *1*
0.  1 0 7, ba 0.625 0.75 0.75 1/3 **0
average:lda-r=0.8 Z=0.675 average network parameters
weighted average:lda-r=0.8 Z=0.708333 weighted according to the proportion of cell outputs
```

key to data order: `cell.  wiring(2-0) rule(hex) ld ld-r Z C`

`cell` the cell index.
`wiring` the wiring scheme of the pseudo-neighborhood (see #9.2).
`(2-0)` reminder of the pseudo-neighborhood index, in this case 2,1,0.
`rule` the rule in hex.
`ld` the λ parameter.
`ld-r` the λ ratio.
`Z` the Z parameter.
`x/3` the proportion of canalizing inputs.
`*1* shows which of the k inputs are canalizing (if none this is not shown),`
`i.e.  *1* - input 1 is canalizing, 2** - input 2, *10 - both inputs 1 and 0.`

19.5.2 Printing network data - mixed k

Example of printing network data (to a printer for DOS, to an xterm for UNIX, or to a file) for a network with mixed rules and non-local wiring, and mixed k.

```
data decode
k=mixed(9-1), n=10 range of neighborhood size k, network size n.
cell.  k, wiring rule(hex) ld ld-r Z reminder of data order (see below).
------------------------------------------
9.  1, - - - - - - - - 0, 01 0.5 1 1 1/1 cell index, k, wiring, rule in hex, parameters.
8.  7, - - 2 6 9 1 6 8 2, 031aa695959fe3993af16ce2c5c8fbeb 0.539 0.922 0.632 0/7
7.  9, 4 8 1 5 8 3 0 8 0, 4bca639feac84bbd260a8e313b890537e5a33673d80617bd8b2170336d8db
ba4521e650efb3d6f5359b805915154bb2fcb5abe4e5dd99745344be450e1abf648 0.51 0.98 0.583 0/9
6.  7, - - 8 1 2 7 6 6 7, 0a4325a68cc996bf4f15a2215e28c3ad 0.461 0.922 0.618 0/7
5.  6, - - - 9 7 8 6 3 1, d29bb9542b547dc2 0.516 0.969 0.619 0/6
4.  4, - - - - - 0 0 9 6, 0f0b 0.438 0.875 0.781 1/4 *2**
3.  2, - - - - - - - 8 8, 0d 0.75 0.5 0.5 2/2 10
2.  1, - - - - - - - - 9, 03 1 0 0 1/1
1.  6, - - - 3 5 0 1 7 5, 3ebd9bd712b3bc8f 0.609 0.781 0.518 0/6
0.  7, - - 6 7 5 4 7 4 4, 17166a2a6289ef9696114bde7725f0c1 0.492 0.984 0.645 0/7
average:ld-r=0.793359 Z=0.589545 network parameter.
weighted average:ld-r=0.847969 Z=0.608713 average network parameters, weighted
according to the proportion of cell outputs.
```

key to data order: `cell.  k, wiring rule(hex) ld ld-r Z C`

`cell` the cell index.
`k` the neighborhood size k.
`wiring` the wiring scheme of the pseudo-neighborhood (see #9.2).
`rule` the rule in hex, the length of the hex string depends on k.
`ld, ld-r, Z, C` same as the first example above (#19.5.1.

19.6 Wiring/rulemix file names

The binary files defining the wiring scheme, rule scheme or wiring/rule scheme combined are named as follows (and encoded as described in 19.7),

	extention	*default file name*
Wiring scheme only:	`.w_s`	`my_wso.w_s`
Rule scheme only:	`.r_s`	`my_rso.w_s`
Wiring + Rule scheme:	`.wrs`	`my_wrs.wrs`

19.7 Wiring/rulemix encoding

The encoding for all three file types above (section 19.6) starts with 5 bytes as follows, where k_{max} = the (maximum) neighborhood size in a mixed-k network (for homogenious k, $k_{max} = k$, k=0, section 8.2), n is the network size, (i, j) the axes of a 2d network, and (i, j, h) for 3d,

byte 0 ...	the lower 7 bits=k or max-k for a k-mix.
	the highest bit=1 indicates a k-mix,
	the highest bit=0 indicates a homogeneous k.
byte 1,2 ...	network size, n.
byte 3,4 ...	i, j.
	if $i = 0$ and $j = 0$ the network is 1d.
	if both $i > 1$ and $j > 1$ the network can be either 2d or 3d.
	$h = n/(i \times j)$. If $h > 1$ the netork is 3d, otherwise 2d.

The encoding continues as follows, for each successive cell index from $0 \ldots (n-1)$,

file type ...	*bytes required*
A. Rules only ...	$2^{k_{max}}$ bytes, (minimum 1 byte),
	starting with the least significant bit in the rule table.
B. Wiring only ...	k_{max} bytes if $n < 255$, $2 \times k_{max}$ bytes if $n \geq 255$,
	indexed $0 \ldots k_{max} - 1$ starting with wiring index 0.
Both Wiring and Rules ...	The combined wiring-rule scheme,
	A followed by **B**.

19.7.1 Mixed-k encoding

The k-mix can be saved in a `.mix` file (see Filing, chapter 35), which is encoded as a binary file of a char array, recording the neighborhood size (k) for each cell (index 0 to n-1) in each byte.

For mixed-k networks, where the actual k of a cell, k_a is smaller than k_{max}, there will be excess bytes for both (A) rules, and (B) wiring, in the encoding described in section 19.7 above, to fill up empty slots. The empty slots are filled as follows,

file type ...	*bytes required*
A. Rules ...	empty bytes are filled with 0s after the rule.
B. Wiring ...	empty slots are filled after the wiring, with
	255 (hex ff), if $n < 255$
	65535 (hex $ffff$), If $n \geq 255$.

Filling empty slots with ff or $ffff$ is how the file keeps track of the k-mix. Note that k=0 is illegal, but an effective k=0 can be set (see section 8.2 and 13.2).

The k-mix can also be saved in a seperate .mix file (see Filing, chapter 35).

19.8 Loading networks and sub-networks

To load a network file as described in sections 19.1, 19.2 and 19.3, a network of the correct type with appropriate parameters must be set up first, the base network or dummy network. In general, the file network must fit into the base network. A file network that is smaller than the base network can be loaded at a set position.

These issues relate to how memory is allocated to different types of network. The constraints are summarized below,

19.8.1 network size

1d . . . $n_{base} \geq n_{file}$,
2d . . . $i_{base} \geq i_{file}$ and $j_{base} \geq j_{file}$
3d . . . $j_{base} \geq j_{file}$, $j_{base} \geq j_{file}$ and $h_{base} \geq h_{file}$

19.8.2 size of k

For mixed k base networks . . . $k_{max,base} \geq k_{max,file}$
For uniform k base networks . . . $k_{base} = k_{file}$

19.8.3 other constraints

wiring+rules . . . To load a combined rule+wiring scheme the base network must have mixed rules and non-local wiring.

For homogemious k networks only

rules only . . . To load a rule scheme the base network must have mixed rules.
wiring only . . . To load a wiring scheme the base network must have non-local wiring.

If the base network has the wrong parameters, various error messages will be displayed and the file will not be loaded. However, if necessary, its easy to trick DDLab by creating simple base networks with more complicated parameters. A mixed k network can be set up with uniform k (see section 8.9). Non-local wiring can be set up with regular 1d, 2d or 3d wiring (see section 11.4.1). Regular 2d and 3d is treated as non-local by default. A mixed rule network can be assigned just one rule (see section 12.6). A mixed k network will be treated as having non-local wiring and mixed rules, even if the "k-mix" is set to uniform k, and all rules are the same. This allows setting up a regular base network, for example a 1d, 2d or 3d CA, into which a random Boolean sub-network can be inserted.

19.8.4 Loading sub-networks in a set position

A file sub-network that is smaller than the base network, which can "fit into" the base network, can be loaded in a set position. Preferably the dimensions (1d, 2d or 3d) of the file and base network should be the same, but if not, the file can still be loaded. The following rules apply,

1. For a 1d file, size n, to be loaded into a 2d or 3d base network, width i, provided $n \leq i$, the 1d file will be converted to a 2d or 3d file. Otherwise the file will be loaded as if both file and base were 1d networks.

2. For a 2d file, $i_2 \times j_2$, to be loaded into a 3d base network, $i \times j \times h$, as long as $i_2 \leq i$ and $j_2 \leq j$, the 2d file will be converted to a 3d file.

Prompts allow the file sub-network to be positioned at a particular place in the base network for either 1d, 2d, or 3d, for example,

<u>*for 1d networks*</u>
1d: array length=150, length file=14, enter start pos *(for example)*
(def 68, max 136): *(the default is a central position)*

<u>*for 2d networks*</u>
2d:i,j=66,66, file: i,j=10,10, enter start coords *(for example)*
(def 28,28, max 56,56) i: j: *(the default is a central position)*

<u>*for 3d networks*</u>
3d:i,j,h=40,40,40, file:i,j,h=9,9,9, enter start coords *(for example)*
(def 15,15,15 max 31,31,31) i: j: h: *(the default is a central position)*

Enter the start coordinates for positioning index 0 of the file sub-network, or enter **return** for the default central position. Any number of independent sub-networks can be loaded into a larger network. Sub-network can be wired into other sub-networks or other parts of the network as described in chapter 17.

A file bigger than the base network (on any axis for 2d or 3d) will be rejected with an appropriate error message, for example,

net size 12<15, cont-ret: *(for a 1d network)*
2d net width 15<20, cont-ret: *(for a 2d network)*
3d net height 45<50, cont-ret: *(for a 3d network)*

However, 1d file networks, size n_{file} can be loaded into 2d or 3d networks, size n_{base}, if $n_{base} \geq n_{file}$, irrespective of the 2d and 3d axes.

19.8.5 Loading mixed k networks

A file network with mixed k can be loaded into a homogenious k or a mixed-k base network provided that the base k or max-k is at least as big as the file k or max-k (the precise base k-mix is not relevant). A base max-k smaller than the file max-k gives the following error message,

mixed k, but max net k3 smaller than max file k5 *(for example)*

The max-k in a mixed k file network may be smaller than max-k in the base network. Max-k can be reduced to the actual maximum k found in the network as described in section 18.10.

Note that only the combined network file, wiring and rules, can be loaded into a base network with mixed k. The combined file need not be a mixed k file. To load "just rules" or "just wiring" in sections 19.1-19.3, the base network must have homogeneous k.

Chapter 20

The network graph, and attractor meta-graph

DDLab includes powerful graph tools which have two distinct applications. Firstly to represent the CA or RBN network itself - the "network graph". Secondly to represent the probability of jumping between basins of attraction due to perturbations to attractor states - the "meta-graph" of the basin of attraction field. As the graph manipulation options are the same for the two applications they are described together in this chapter.

The network graph does not allow changes to the underlying network (see chapter 17 to do this), but for both the network graph and attractor meta-graph, very flexible methods are available for rearranging and unraveling the graph, including dragging vertices and defined components to new positions with "elastic band" edges, rescaling nodes and links, and changing connections just within the graph. An "ant" can be launched into the network that moves according to the link probabilities (as in a Markov chain) keeping a count of node hits.

The data from which the graph is generated can be shown as a table, called the "adjacency matrix" for networks, and the "jump-table" for meta-graphs.

20.1 Unraveling the meta-graph and the network graph

The same methods apply for rearranging and unraveling the network graph and meta-graph. The network graph represents the actual RBN or CA network, how the network elements are linked by their wiring, and should not be confused with the attractor meta-graph, where the nodes represent basins of attraction, and links represent jumps between basins.

Single nodes, connected fragments, or whole components, can be dragged with the mouse, according to inputs and/or outputs. The distance of fragment links from a node can be restricted, i.e. dragging the node + its immediate link nodes (step 1), the node + immediate links + their immediate links (step 2), etc. Arbitrary 1d, 2d and 3d blocks can also be dragged. Nodes with the fewest links can be automatically moved to the outer edges. This makes it easy to unravel a graph.

The pre-programmed graph layouts available are a circle of nodes, a spiral, or in 1d, 2d or 3d. The graph can be rotated, expanded, contracted, and various other manipulations can be performed. The graph layout can be saved/loaded (a `.grh` file). The graph is displayed in a large window across the lower part of the screen, smaller for the meta-graph to allow space to show the basin of attraction field above the window.

The default presentation, as in figure 20.1(*left*), shows the vertices or nodes arranged in a circle. By default the sizes of nodes are scaled according to the neighborhood size k (network graph), or basin volume (meta-graph), and the thickness of edges is scaled according to both this and the fraction of available outputs represented by the link. If a node has connections to itself, this is represented by a short stub projecting from the node, also scaled as above. The nodes can also be made the same size, and the links can be represented by thin lines with arrows.

The nodes are numbered (if big enough) according to the cell order, 0 to n-1 (network graph), or basin number starting with 1 (meta-graph). Successive nodes are assigned different colors cycling through six colors. Outgoing edges from a node are the same color as the parent node, and aligned asymmetrically clockwise relative to the source node. This ensures that outgoing and incoming edges do not overlap if two nodes link to each other. The directed edges are separated by a central gap.

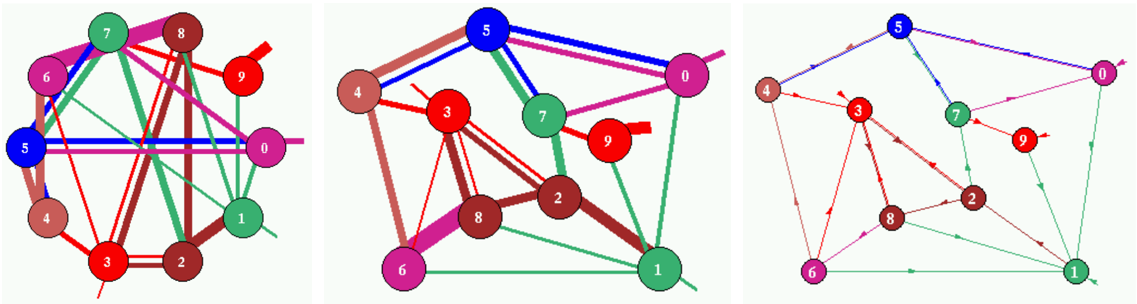


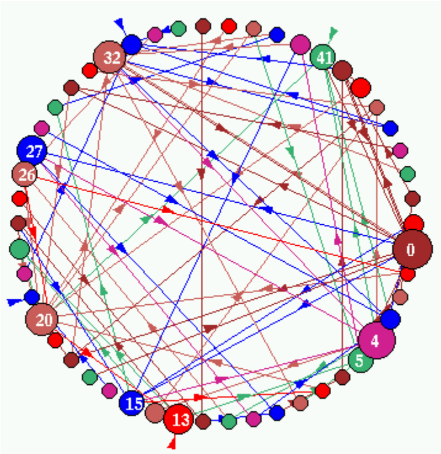
Figure 20.1: The network graph for an RBN with uniform k. The network (n=10, k=3) is defined figure 20. *left*: the default graph with nodes arranged in a circle. *center*: the graph rearranged by dragging nodes with the mouse. *right*: edges shown with arrows.

	0	1	2	3	4	5	6	7	8	9	out	k
0:	1	.	.	.	.	1	.	1	.	.	3	3
1:	1	1	.	.	.	.	1	.	1	1	5	3
2:	.	2	.	1	.	.	.	.	.	.	3	3
3:	.	.	1	1	2	.	1	.	1	.	6	3
4:	.	.	.	.	.	1	1	.	.	.	2	3
5:	1	.	.	.	1	.	.	1	.	.	3	3
6:	.	.	.	.	.	.	.	.	1	.	1	3
7:	.	.	1	.	.	1	.	.	.	.	2	3
8:	.	.	1	1	.	.	.	.	.	.	2	3
9:	.	.	.	.	.	.	.	1	.	2	3	3

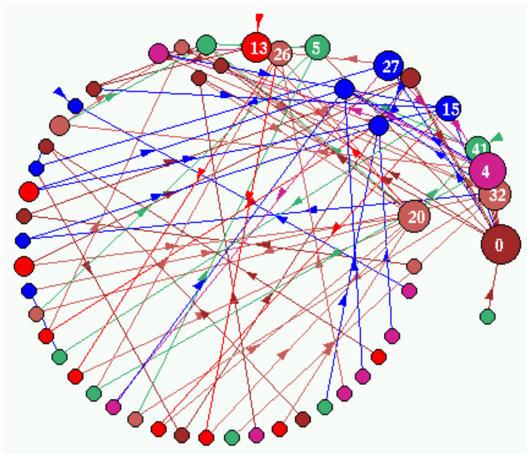
Figure 20.2: The adjacency matrix of the network graph for the RBN in figure 20.1. The columns (0 to n-1) show the number of output wires from each row cell (0 to n-1) to other cells. This can also be shown as fractions of total outputs. The last two columns show the total outputs and inputs (k), for each cell.

20.2 The network graph

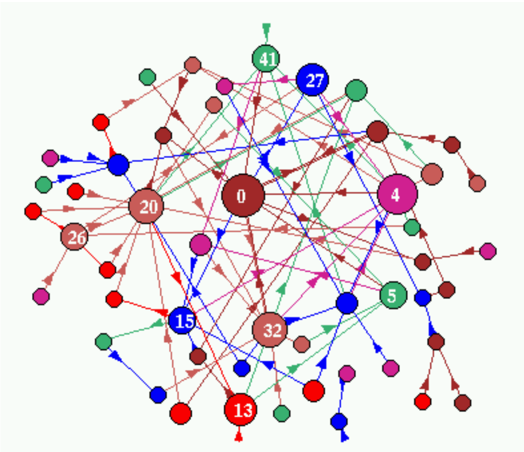
The network graph is selected from the (top right) network architecture prompt, which is displayed at various stages in DDLab, firstly after special wiring is set (chapter 11), secondly after both the wiring and rules have been set, in the main sequence of prompts. The prompt can also be selected at later stages in DDLab, while attractor basins (sections 30.4, 30.5), or space-time patterns (section 32.15) are being drawn. Section 32.15 explains how space-time patterns can be shown within the network graph layout.



the default graph with nodes arranged in a circle



the circle layout, nodes ranked by k, and pulled apart by dragging node 0 + input and output nodes 1 step away



the graph rearranged by automatically unscrambling a ranked spiral layout, then adjusting single nodes

Figure 20.3: The graph of a scale free network (n=50) with a power-law distribution of both k (inputs), and outputs. To set up such a network see sections 8.8.2 and 17.8.4. Scale free topology characterizes many natural and artificial networks, from the Internet to metabolic networks [1]. The figures illustrate how the graph can be unraveled. Links are shown with arrows, Nodes are scaled according to k.

Enter **g** at the network architecture prompt in section 17.1 to show the network as a graph, or as a table (the adjacencey matrix). At the same time, the following top right network graph options are displayed,

NETWORK graph: drag-(def), ant-a rank-k unscram-u
settings-S rotate-x/X flip-h/v, exp/contr: nodes-e/c links-E/C both-B/b
table-t/T Unreach-U, tog: nodes-n links-l window-w
layout: file-f circle/spiral-o/O 1d/2d/3d-1/2/3 rnd-r/R quit-q:

For a large network the following top right message may be displayed first, with an inset window that monitors progress such as $\boxed{\textbf{55\%}}$,

computing network graph, 1600 nodes, quit-q, or wait...

The network graph is shown in a large window below the prompt, occupying most of the screen.

20.3 The meta-graph of the basin of attraction field

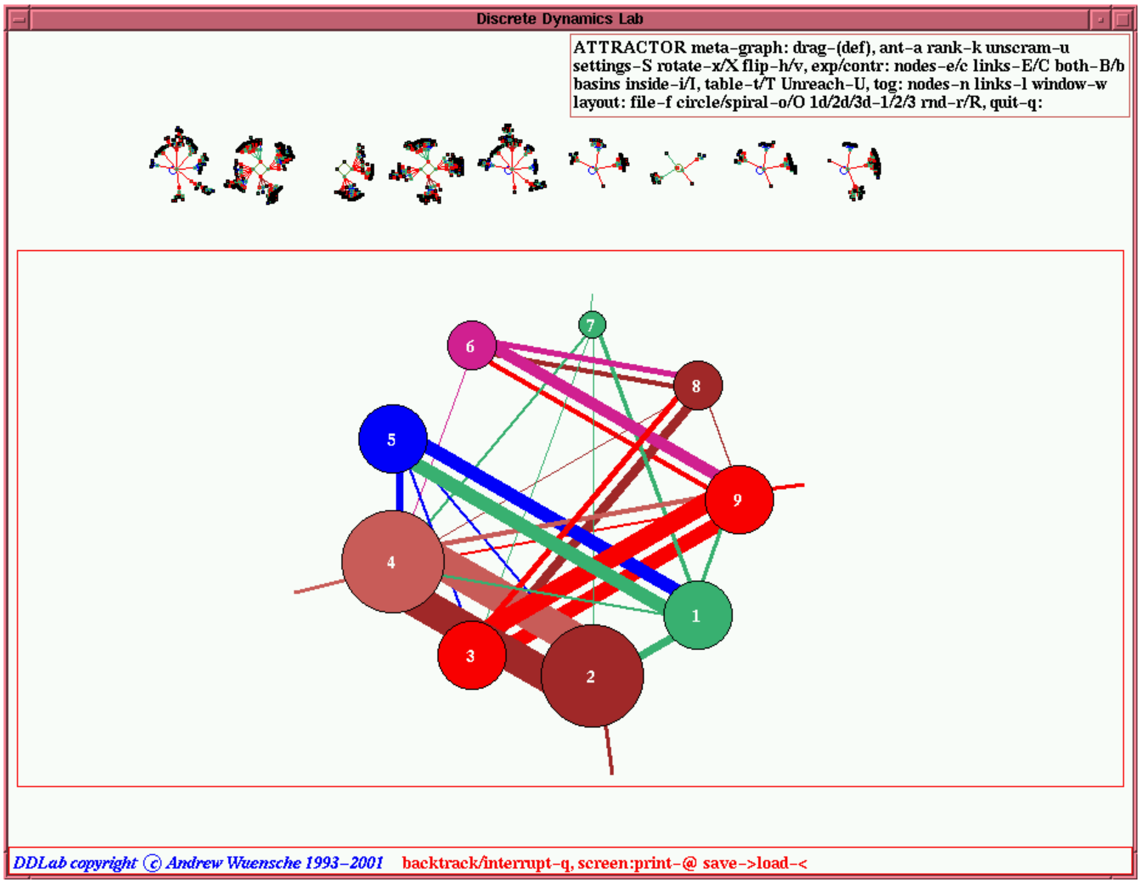


Figure 20.4: The meta-graph of a RBN, n=10, k=3 (as defined in figure 20).The basin of attraction field, rescaled to a small size at the top of the screen, is generated first, its scale and layout amended as described in section 20.3.2. The meta-graph itself is displayed in a large window across the lower part of the screen. Graph prompts are shown in a top right window.

Perturbations due to noise or external signals are most likely to take affect once the system has relaxed to its attractor, because thats where the dynamics spends the most time. Taking 1-bit perturbations to attractor states as the simplest case, the meta-graph represents the probabilities of jumping between basins, and gives some insight into the stability and adaptability of the dynamics, which is especially relevant in attractor models of memory in neural networks and of cell differentiation in genetic networks.

The meta-graph algorithm analyzes the basin of attraction field to track where all possible 1-bit flips to attractor states end up, whether to the same, or to which other basin. The information is presented in two ways, as a jump-table: a matrix showing the jump probabilities between basins, and as a meta-graph: a graph with weighed vertices and edges giving a graphic representation of the jump-table.

		2.	1.	0.	rule(hex)
3	9..	9	9	1	81
2	8..	3	1	6	9e
2	7..	9	0	5	7b
1	6..	4	3	1	1f
3	5..	4	0	7	21
2	4..	3	3	5	77
6	3..	8	3	2	c7
3	2..	8	7	3	47
5	1..	2	1	2	c7
3	0..	5	0	1	c8

Figure 20.5: The RBN used in some of the meta-graph examples, and for the network graph in figures 20.1 20.2, $n = 10$, k=3. This network "matrix" is generated as described in section 17.2.

The attractor meta-graph algorithm works for any network (with either synchronous or sequential updating), for CA (with compression suppressed), for RBN, and also for random maps, and applies to any reverse algorithm including exhaustive testing.

Though mainly applied when drawing the complete basin of attraction field, the meta-graph can also be computed for the attractor histogram (see sections 31.17 - 31.18.2), though this may take a long time for a histogram with many long period attractors. The attractor histogram is applied to find data on attractors for large networks, especially RBN, where it may be not be possible, or practical, to generate the basin of attraction field, (see section 1.6). DDLab provides statistical methods for gathering data on the range of attractors and their relative sizes. This is done by running a network forwards from many random initial states, identifying different attractor "types", and creating a histogram of the frequency of each type.

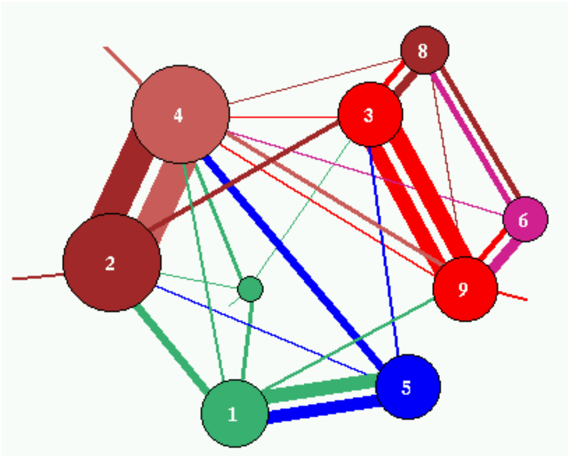


Figure 20.6: Dragging nodes to new positions with the left mouse button. This is the meta-graph of the RBN defined in figure 20.4.

20.3.1 Selecting the meta-graph

The meta-graph prompt belongs to the **pause/data** category (chapter 27) of "output parameter" prompts. To go directly to this category, enter **t** at the first "output parameter" prompt in section 25.2, or arrive there by viewing the output parameters in sequence. After other prompts in section 27.5, the following prompt is presented,

attractor meta-graph -m:

Enter **m** to show the meta-graph. For CA, if "compression" is active, it will be turned off (see section 26.1), and the following message will be displayed,

... compression in basin/fields turned OFF.

If **m** is selected, a further prompt is presented to display the jump table in the xterm window (not for DOS). The table can also be displayed in the meta-graph window within DDLab (see section 20.10).

show jump-table: numbers-n fractions-f both-b: *(not for DOS)*

Enter **n** for the numbers of jumps, **f** for the fractions of jumps, or **b** for both.

If the meta-graph option is selected, the basin of attraction field is generated first, then DDLab automatically generates the meta-graph. The meta-graph can also be selected in section 31.18.2 for the attractor histogram. The following top right graph options are presented,

ATTRACTOR meta-graph: drag-(def), ant-a rank-k unscram-u
settings-S rotate-x/X flip-h/v, exp/contr: nodes-e/c links-E/C both-B/b
basins inside-i/I, table-t/T Unreach-U, tog: nodes-n links-l window-w
layout: file-f circle/spiral-o/O 1d/2d/3d-1/2/3 rnd-r/R, quit-q:

The meta-graph may take some time to compute if the basin of attraction field (or attractor histogram) has many long period attractors. The following top right message may be displayed first, with an inset window that monitors progress such as $\boxed{\textbf{55\%}}$,

computing meta-graph, 25 basins, quit-q, or wait...

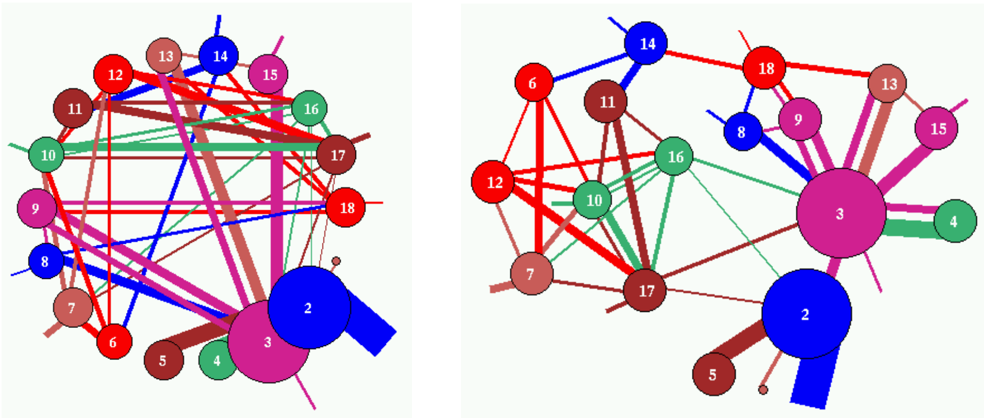


Figure 20.7: A meta-graph with nodes arranged in a circle, the default (left), and rearranged with the mouse to group strongly linked components (right). This example is for a 1d CA, n=10, k=3, rule 141.

20.3.2 Amend basin layout for the meta-graph

If selected, DDLab automatically generates the meta-graph in a window occupying the lower 2/3 of the screen as in figure 20.4, after the basin of attraction field is drawn. It a good idea to reposition the basins near the top of the screen so they are not obscured by this window, and also to rescale the basin size so that basins will eventually fit within the nodes of the meta-graph (see section

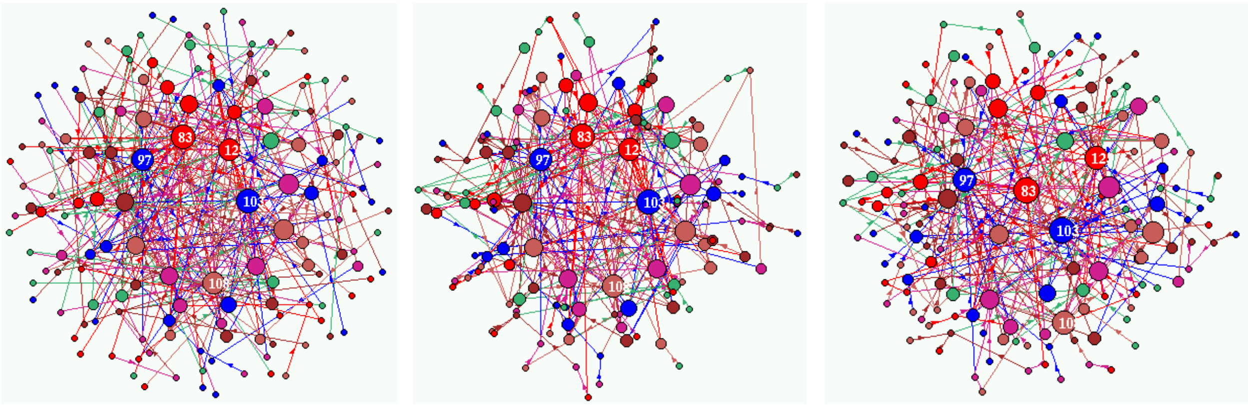
20.7). This is done from the **layout** category of prompts described in chapter 25. Enter **l** at the first "output parameter" prompt in section 24.1, then (for example) enter **2** (section 25.2) to rescale basins to a small size, then change the **y** layout setting to **70** (section 25.2.1) to draw basins near the top of the screen.

In any event, the meta-graph window can be togged on/of to show any basins hidden behind it.

20.3.3 Amend mutation for the meta-graph

You might want to see a sequence of basin of attraction fields and their meta-graphs. Select the required change (mutation) to the network for successive fields in chapter 28.

20.4 Initial graph options



show as a spiral, and rank | automatically unscramble | shake, then drag

Figure 20.8: One possible method of unscrambling a scale free RBN, n=150. *left*: enter **O** to show the network as a spiral, then **k** to rank the nodes, placing nodes with the highest k at the center. *center*: enter **u** to automatically unscramble the layout, possibly more than once. *right*: enter **r** to "shake" the layout, then make final adjustments by dragging single nodes or fragments. The links were first changed to thin lines with arrows (enter **l**), and the arrows size was reduced (enter **S** for settings).

A summary of the initial graph options in section 20.2 and 20.3.1 are given below. There is also a second set of options related to dragging nodes and fragments summarized in section 20.5. Further sections and figures in this chapter describe some of the options in greater detail. The options are the same for the network graph and meta-graph, except for **basins inside-i/I** which applies only to the meta-graph. The key presses take effect immediately, without entering **return**.

drag-(def) ... enter **return**, or click the left or right mouse button, for a further set of "drag node" options presented in a top right window. These include dragging nodes, or fragments of the graph, to new positions with the left mouse button, with "elastic band" edges. The fragments can be connected components defined as being at a given distance from a node according to input links, output links or either (i.e. any link), or can be arbitrarily defined as blocks in 1d, 2d, or 3d. The fragments can be separately dragged, flipped and rotated. (see section 20.5).

ant-a ... enter **a** to launch a projectile or probabilistic "ant" in the graph to trace a (Markov chain) path according to the link probabilities, keeping track of the frequency of visiting vertices (see section 20.6).

rank-k ... enter **k** to rank the nodes by reordering node positions in the graph according to inputs (k) for the network graph, or basin volume for the meta-graph.

unscram-u ... enter **u** to automatically unscramble the graph, placing weakly connected nodes on the outer edges, close their connections. This works best in the circle or spiral layout.

settings-S ... enter **S** to revise the default settings for the rotation angle, the scaling factors for nodes and links, and for the arrow size of arrows that can be shown on the links (see section 20.8).

rotate-x/X ... enter **x** or **X** to rotate the graph clockwise or anti-clockwise (about the window center) by the default angle of 15 degrees, or a revised angle.

rotate-h/v ... enter **h** or **v** to flip the graph horizontally or vertically.

nodes-e/c ... enter **e** or **c** to expand or contract the size of nodes and thickness of links by a default factor (1.2), or a revised factor.

links-E/C ... enter **E** or **C** to expand or contract the length of links by a default factor (1.1), or a revised factor.

both-B/b ... enter **B** or **b** to expand or contract both nodes and links at the same time by the default factors above.

basins inside-i/I ... (*meta-graph only*) enter **i** or **I** to redraw the basins of attraction at the meta-graph nodes (see section 20.7). **i** keeps the graph. **I** shows just the basins, so is an alternative method of presenting the basin of attraction field with a specific layout, see section 27.8.

table-t/T ... enter **t** or **T** to show the adjacency matrix (network graph) or the jump table (meta-graph), for the numbers of links **t**, or the fractions total links **T**, between each ordered pair of nodes, (see section 20.10).

Unreach-U ... enter **U** to identify and separate unreachable or hard to reach nodes in the graph, (see section 20.9).

nodes-n ... enter **n** to toggle between scaled and unscaled nodes and edges.

links-l ... enter **l** to toggle between scaled links and links shown as thin lines with arrows (see figure 20.1).

window-w ... enter **w** to toggle the graph window. This might be useful in case it covers the basin of attraction field.

layout: file-f ... save or load the current graph layout of node positions, a `.grh` file (see Filing, chapter 35).

circle-o/O ... enter **o** or **O** to show the graph arranged in a circle or spiral. The spiral option needs about 30 nodes or more to work properly.

1d/2d/3d-1/2/3 ... enter **1**, **2** or **3** to show the graph arranged in 1d, 2d or 3d. This is especially effective for 2d and 3d networks.

rnd-r/R ... enter **r** to "shake" the layout, repositioning nodes close to their current position, at random. Enter **R** for a completely random layout.

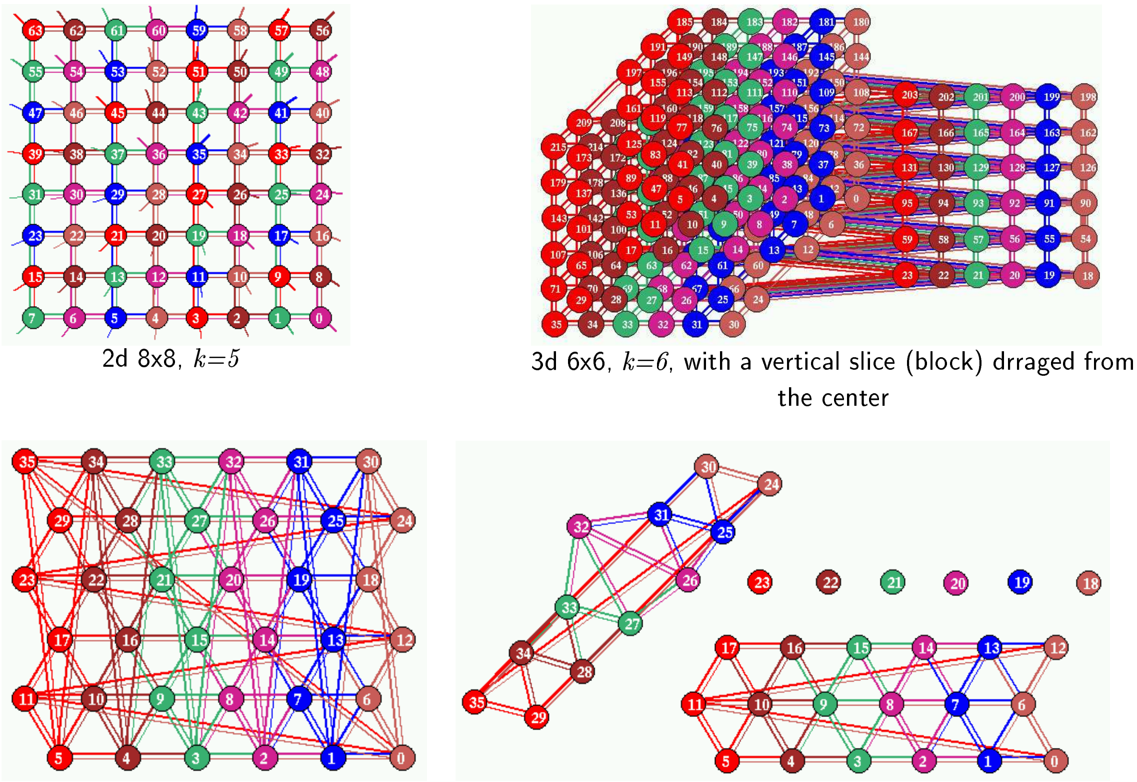
quit-q ... enter **q** to quit the graph and return to the prompt in section 17.1.

20.5 Dragging nodes or fragments

Enter **return**, or click the left or right mouse button in the initial graph options (section 20.4), to enable dragging nodes or fragments with the mouse, and other features.

The initial setup allows single nodes to be dragged. A node is activated and dragged by clicking and holding down the left mouse button, releasing the node in a new position. While dragging, the links will act as "elastic bands". Sometimes a node will not activate, if so, click on it with the right mouse button, then try again with the left. When a node is activated its number appears in the prompt (**single node 5**, for example), and also in a small inset in the top right hand corner of the prompt window, **cell 5** for example.

As well as dragging a single node, there are options for dragging connected fragments, or separate components of the graph. The fragments can be defined as being at a given distance from a node according to input links, output links or both, or can be arbitrarily defined as blocks in 1d, 2d, or 3d.



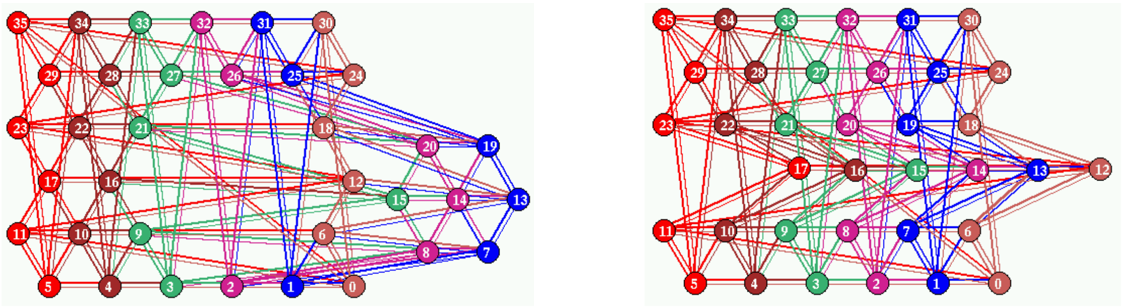
2d 8x8, k=*5*

3d 6x6, k=*6*, with a vertical slice (block) drraged from the center

2d 6x6, k=*6*

network broken by disconnecting some nodes, a component was dragged and rotated

Figure 20.9: Graphs of a 2d and 3d CA. *top left*: a 2d CA. *top right*: a 3d CA, an axonometric projection seen from below as if looking up at the inside of a cage. A vertical slice has been defined and dragged from the graph. *bottom left*: a 2d CA where the links follow a triangular grid (see section 9.4). The graph was first presented in 2d, then *bottom right*: broken into separate components by disconnecting some nodes. One component was dragged to a new position and rotated. Note that breaking, and creating, new connections affects only the graph, not the underlying network which can be restored.



dragging node 14 and its step=1 inputs | dragging a block, nodes 18-23

Figure 20.10: Graphs of a 2d, 6×6 CA, k=6, as in figure 20.9, but with links shown as lines with arrows, showing dragging a connected fragment and a arbitrarily defined block.

Components and fragments can be separately dragged, rotated and flipped. Single nodes can have their inputs and/or outputs links cut. Links between pairs of nodes can be cut or added. So called "gap nodes", which have only inputs, or only outputs, can be disconnected from the graph, showing the effective components, because gap nodes cannot transmit information. Note that breaking and creating new connections affects only the graph, not the underlying network which can be restored. The initial prompt, for a network graph, is as follows,

single node 0: drag/release - left mouse button *(***node 1** *for the meta-graph)*
not active?-click right button first, rotate-x/X flip-h/v gap nodes-g
links: restore net-n, block-B, i/o/b to activate
step-(1-9) nolimit-0, single-s in/out/either-i/o/e all-a exit-q:

The prompt changes if **i**, **o** or **e** (for inputs, outputs, or either are entered, for example, entering **e**, and clicking node 19, then node 23 (for the network in figure 20.10), gave the following,

in-out node 23, step=nolimit: drag/release - left mouse button **node 23**
not active?-click right button first, rotate-x/X flip-h/v gap nodes-g
links 23: cut/restore/net-c/r/n, link 23-19: cut/add/restore-C/A/R
step-(1-9) nolimit-0, single-s in/out/either-i/o/e all-a exit-q:

This indicates that the last node clicked was 23; links to that node can be cut. The previous node was 19; **link 23-19** appears in the prompt, links between 23 and 19 can be cut or added. When adding an input link direction will point from 23 to 19. When adding an output link the direction will point from 19 to 23. **step=nolimit** signifies that dragging a node will also drag the connected component, indirectly linked by either inputs and outputs, along with it. This can be changed by entering a number **step-(1-9)**, to limit the size of the linked fragment. For example, entering 1 results in **step=1** in the top line, and only the nodes immediately linked to the active node will be dragged.

Enter **s** to revert to dragging single nodes. The option **block-B**, to define an arbitrary block to be dragged, is only available if **single node** is active. If **B** is entered, the block is first defined in 1d, 2d or 3d (see section 20.5.1). The default block is the block between the last 2 mouse clicks. Then the first line of the prompt reads as follows (for example),

node 23, Block: drag/release - left mouse button **node 23, 2d Block**

The defined block can the be independently dragged. The following is a summary of the drag options, which take effect as soon as the key is pressed, without entering **return**,

rotate-x/X ... enter **x** or **X** to rotate the graph or fragment 15 degrees clockwise, or anticlockwise. Rotation is centered on the active node. The default rotation angle can be revised in section 20.8).

flip-h/v ... enter **h** or **v** to flip the graph or fragment horizontally or vertically.

gap-g ... enter **g** to cut all links to "gap nodes". Gap nodes have only inputs, or only outputs. Disconnecting them from the graph will show the effective components, because gap nodes cannot transmit information.

if **single node** *is active*

restore net-n ... enter **n** to restore the graph to the underlying network or meta-graph.

block-B ... enter **B** to define a block. The prompt heading changes to **node 23, Block:** (for example). For a 2d or 3d network graph, a block can be defined in 2d or 3d according to its 2 outer corners, similar to the methods in sections 17.5.4 and 17.6.4. For a meta-graph or a 1d network graph, the block is simply a range of nodes between an upper and lower limit (as in section 17.4.4). In general the outer edges of the default block are defined by the last two nodes that were activated with the mouse, but prompts are displayed to set the block by hand, described in section 20.5.1.

if **inputs, outputs** *or* **in-out** *is active*

links 23: cut/restore/net-c/r/n ... *for example*

enter **c** to disconnect, **r** to restore, the active node from/to the network by cutting/reconnecting all its input, output, or any link, depending on which option, **inputs, outputs** or **in-out** is active.

net-n ... enter **n** to restore the graph to the underlying network or meta-graph.

link 23-19: cut/add/restore-C/A/R *for example*

cut-C ... enter **C** to cut, **A** to add, **R** to restore, links between the active node and the previously active node. The node numbers appear in the prompt. The links that are cut/added/restored are either input, output, or any link depending on which option, **inputs, outputs** or **in-out** is active.

net-n ... enter **n** to restore the graph to the underlying network or meta-graph.

step-(1-9) ... enter a number between **1** and **9** to limit a fragment by the distance from the active node. This relates to a continuous chain of directed edges, either inputs to the node, outputs from the node, or either inputs or outputs. For example, enter **1** to limit the fragment to immediate links, **2** to include indirect links 2 steps away, etc., (up to 9 steps). The current status is shown in the prompt heading, for example **step=1**

nolimit-0 ... enter **0** to define a fragment by a continuous chain of inputs, outputs, or either inputs and outputs, however indirectly, relative to the active node. Gap nodes would interrupt such a chain. The current prompt heading will show **step=nolimit**

single-s ... enter **s** to restore dragging single nodes. The heading changes to **single node 23:** (for example).

in/out/either-i/o/b ... enter **i**, **o** or **e** to drag the active node plus other nodes that have inputs, output, or either (i.e. any link) to it (directly or indirectly), according to the current **step** setting. The prompt heading changes to **inputs node 5**, **outputs node 5**, or **both in-out node 5** (for example).

all-a ... enter **a** to drag all nodes, the whole graph. The prompt heading changes to **all nodes:**.

exit-q ... Enter **q** to exit and return to the prompts in section 20.4.

20.5.1 Defining a block

If **single node** is active, enter **B** in section 20.5 to define a block. The following top right prompts are presented, depending if the type of graph,

for a 1d network n=62, or a metagraph
define 1d block:
1d block (0-61, def 49-59), *(values shown are examples)*
low:
high:

for a 2d network, 6x6
2d network: define block: 1d-1, 2d(def):
2d block (this=0,2 max=5,5), *(values shown are examples)*
low corner (def 0,2) i: j:
high corner (def 5,2) i: j:

for a 3d network, 6x6x6
define 3d network 1d-1 3d-(def):
3d block (this=0,3,0 max=5,5,5), *(values shown are examples)*
low corner (def 0,3,0) i: j: h:
high corner (def 5,3,5) i: j: h:

A 1d block is defined by entering the low and high node index. This also applies for a 2d or 3d network if **1** is selected first. For a 2d or 3d network, enter the coordinates of the low and high corner to define the block.

In general the default values correspond to the last two nodes that were activated with the mouse, and these values are shown in the prompt. Enter **return** to accept each default.

Once the block is defined, dragging any node (whether inside or outside the block) will also drag the block, and **rotate-x/X**, **flip-h/v**, will apply just to the block.

20.6 Probabilistic "ant"

Enter **a** in section 20.4 to launch a projectile or probabilistic "ant" in the graph. The ant has a red body and leaves a black trail on the center line between nodes. The default path starts at a randomly selected node and traces a Markov chain, a path that takes the next link according to the output probabilities. The "ant" keeps track of the frequency of visiting vertices. There are a number of further ant options, presented in a top right as follows,

probabilistic ant: quit-q speed-< / > pause-p
restart at random node -r, redraw graph -g
show hits -h, tog: show ant (ON)-a, random statistics(OFF) -s:

The meaning of these prompts is listed below,

quit-q . . . quit the ant routine and return to the the options in section 20.4.

speed-< / > . . . enter < to slow the ant down, or > to speed it up.

restart at random node -r . . . restart the ant at a node selected at random, continue to keep track of node visits.

redraw graph -g . . . redraw the graph to delete the ant trail, continue to keep track of node visits.

show hits -h . . . show the frequency of ant hits, or node visits. See section 20.6.1 for more details.

tog: show ant(ON) . . . toggle the display of the ant and its trail. The current status (ON or OFF) is indicated. If the ant graphics are off, the statistics of hits are gathered much faster.

tog: random ant(OFF) . . . toggle between moving the ant as in a Markov chain (the default), or restarting at random nodes for each step. The current status, random ant ON or OFF, is indicated. See section 20.6.1 for more details.

20.6.1 Show ant hits

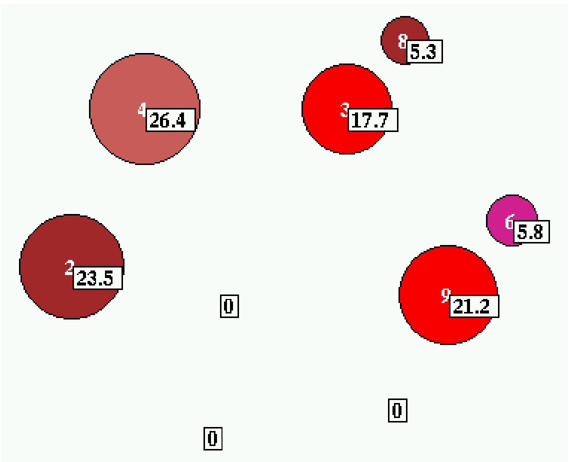


Figure 20.11: Probabilistic ant hits, or node visits (enter 'h' in section 20.6). In this example nodes 7, 1 and 5 have no hits because they are unreachable from nodes 2,4 and 3,8,6,9. This is the meta-graph of the RBN defined in figure 20.4, rearranged as in figure 20.

If **h** is entered while the probabilistic ant is active, the graph is redraw to show the frequency of hits so far. Just the nodes are displayed in their current positions, scaled according to hit frequency (edges are not shown). Nodes with no hits not shown. The following top right reminders and options are presented,

showing % hits on each basin
total hits so far = 1453697 *(for example)*
show % frequency: only-N +nodes-n, redraw-r cont-ret:

Enter **n** to add the frequencies of hits, as a percentage of total hits so far, to the scaled nodes (as in figure 20.6.1). Enter **N** to show just the frequencies without showing the nodes, which might be necessary if large nodes obscure the data. Enter **r** to redraw the original scaled nodes without frequencies. Enter **return** to redraw the metagraph and return to section 20.6, where the ant remains active. Note that if the ant graphics is toggled off in section 20.6 (by entering **a**), statistics of hits will be gathered much faster.

The **random ant** option in section 20.6 produces a different kind of sample; at each step the ant is re-started at a random node, biased by k (network), or by the relative size of the basin of attraction that the node represents (meta-graph) . Thus the ant can hit all reachable nodes even if they occur in separate components (ergodic sets) of the meta-graph.

By contrast, the default ant starts at a random node and follows a Markov chain, a continuous path according to the output probabilities in the graph. Some graphs are fully interlinked, but in a graph consisting of distinct components, if the path starts within such a component, it will be trapped inside. Note that the eigen vectors of the jump table of a component (which may be the whole graph) should be the same as the list of hit frequencies of the component.

20.7 Redraw basins at meta-graph nodes

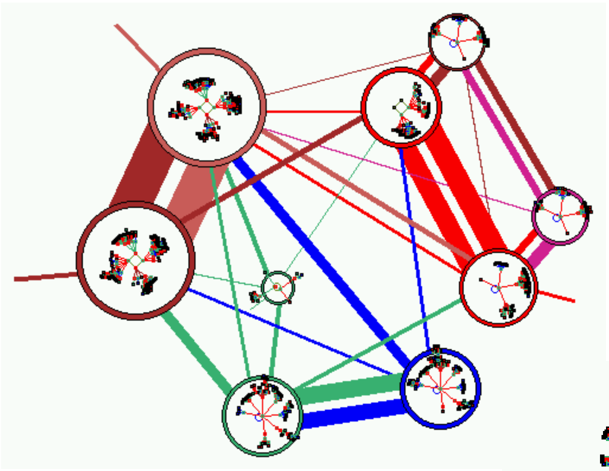


Figure 20.12: Basins of attraction re-drawn at the meta-graph nodes. This is the meta-graph of the RBN defined in figure 20.4, rearranged as in figure 20..

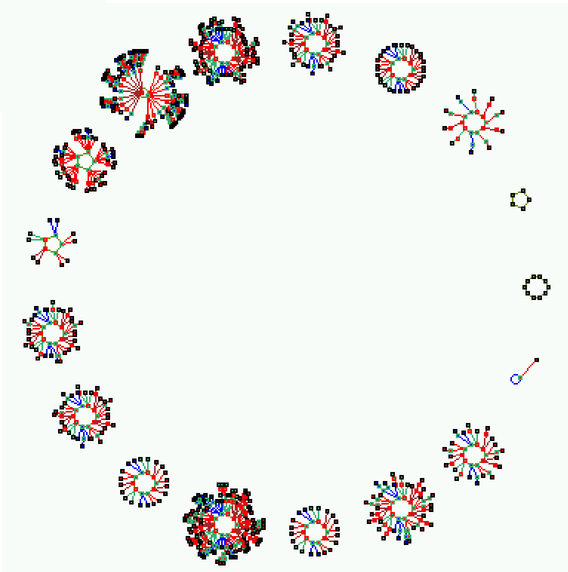


Figure 20.13: Basins of attraction in circle layout, for a CA, rule 277, n=10.

Enter **i** or **I** in section 20.3.1 to redraw the basin of attraction field in the meta-graph, with each basin centered on its assigned node, according to its current layout and presentation.

Enter **i** to keep the graph, as in figure 20.6.1. where basins were drawn at a small scale to fit inside the meta-graph nodes (see section 20.3.2).

Enter **I** to show just the basins at the meta-graph nodes positions, but without the meta-graph, as in figure 20.6.1. This is an alternative method of arranging the basin of attraction field with any arbitrary layout (see section 27.8).

Enter **return** to revert to the options in section 20.4 and redraw the meta-graph.

20.8 Revise settings

Enter **S** in section 20.4 to change the default settings for the rotation angle, and the expand/contract factors for both nodes and links, and the arrow size for links shown as thin lines with arrows, as in figure 20.1.

The following top right prompts are presented in sequence,

> **settings:rotation angle (start=15), now 15.00 degrees:**
> **expand/contract nodes: factor (1 to 2, start=1.2), now 1.20:**
> **expand/contract links: factor (1 to 2, 1.1), now 1.10:: change arrow size: (0 to .3, start=0.07), now 0.07:**

If required, enter the new setting, which may be decimal. The expand/contract factor range is from 1 and 2. The arrow size range is from 0 to 0.3, set this to 0 to suppress arrows. Revised setting become the defaults.

20.9 Unreachable nodes

Enter **U** in section 20.4 to show up unreachable or hard to reach nodes by disconnecting and/or isolating them from the rest of the graph. A basin is unreachable if it has no incoming links from other basins. An unreachability threshold of zero is the default, but this can be reset to a higher value, so that a node is isolated if it has x or fewer incoming links. The following top right prompts are presented,

> **links from unreachable basins: suppress u: restore-(ret):**
> *if* **u** *is entered, the following further prompts are presented*
> **isolate unreachable -i:**
> **change unreachable threshold (now 0):**

Enter **u** above to suppress outgoing links to basins whose incoming links are at or below the current threshold, or **return** to restore these links. If **u** is entered, at the next prompt enter **i** to isolate the unlinked nodes, which will be repositioned randomly on the far left side of the window. At the next prompt enter a new threshold if required.

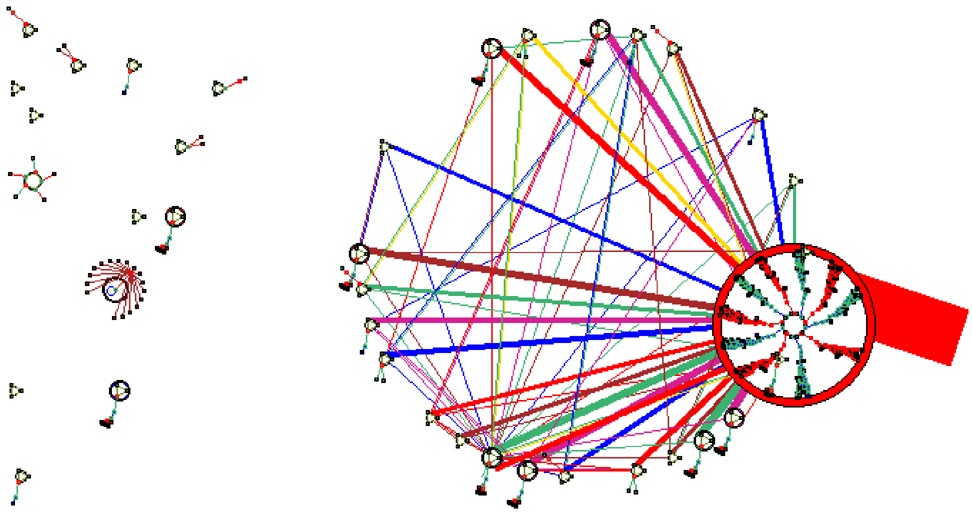


Figure 20.14: Disconnecting and isolating unreachable nodes. The isolated nodes are repositioned randomly on the far left side of the graph window. in this case with the default unreachability threshold of zero. This example is for a metagraph for a 1d CA, n=10, k=3, rule 131. The basins of attraction were redrawn at the nodes (see section 20.7).

20.10 The adjacency matrix and jump-table

Enter **n** or in **f** section 20.4 to show a table of outputs from each node. This is called the "jump-table" for the meta-graph, and the "adjacency matrix" for the network graph. The table can be presented either according to the number of outputs (enter **n**), or the fraction of possible outputs from each node (enter **f**). The table is presented in the graph window.

For a meta-graph jump-table, successive rows are labeled 1 to N, were N is the number of basins in the field. For each basin, the columns, also labeled 1 to N, show the number (or fraction of total) jumps to each basin (see figure 20.15). The three further columns labeled **P**, **J** and **Volume** are explained below,

P ... the period of the attractor.

J ... the total number of possible 1-bit flips or jumps, which equals the network size multiplied by the attractor period, $n \times P$.

Volume ... The total number of states in the basin of attraction, and the percentage of state-space, for example **110=10.74%**.

Note that the jump-table can also be displayed in the xterm window (not for DOS) from the meta-graph prompt in section 20.3.1.

For a network graph adjacency matrix, successive rows are labeled 0 to n-1, were n is the network size. For each network element, the columns, also labeled 0 to n-1, show the number (or fraction of total) outputs to each network element (see figure 20.2). Two further columns headed **out** and **k** show the total outputs and inputs from/to each network element.

	1	2	3	4	5	6	7	8	9	P	J	Volume
1:	.	3	.	1	5	.	.	.	1	1	10	110=10.74%
2:	.	4	4	32	.	.	.	.	.	4	40	235=22.95%
3:	.	.	.	4	.	.	.	8	28	4	40	106=10.35%
4:	.	32	.	4	.	.	.	.	4	4	40	233=22.75%
5:	5	1	1	3	.	.	.	.	.	1	10	108=10.55%
6:	.	.	.	1	.	.	.	3	6	1	10	54= 5.27%
7:	8	2	2	6	.	.	2	.	.	2	20	18= 1.76%
8:	.	.	5	1	.	3	.	.	1	1	10	56= 5.47%
9:	.	.	6	1	.	2	.	.	1	1	10	104=10.16%

	1	2	3	4	5	6	7	8	9	P	J	Volume
1:	.	.300	.	.100	.500	.	.	.	.100	1	10	110=10.74%
2:	.	.100	.100	.800	.	.	.	.	.	4	40	235=22.95%
3:	.	.	.	.100	.	.	.	.200	.700	4	40	106=10.35%
4:	.	.800	.	.100	.	.	.	.	.100	4	40	233=22.75%
5:	.500	.100	.100	.300	.	.	.	.	.	1	10	108=10.55%
6:	.	.	.	.100	.	.	.	.300	.600	1	10	54= 5.27%
7:	.400	.100	.100	.300	.	.	.100	.	.	2	20	18= 1.76%
8:	.	.	.500	.100	.	.300	.	.	.100	1	10	56= 5.47%
9:	.	.	.600	.100	.	.200	.	.	.100	1	10	104=10.16%

Figure 20.15: The jump-table the meta-graph in figure 20.4, according to *top*: the number of jumps, and *below*: the fraction of possible jumps (bottom), from each attractor. Zero values are show as dots.

20.10.1 Scanning large tables

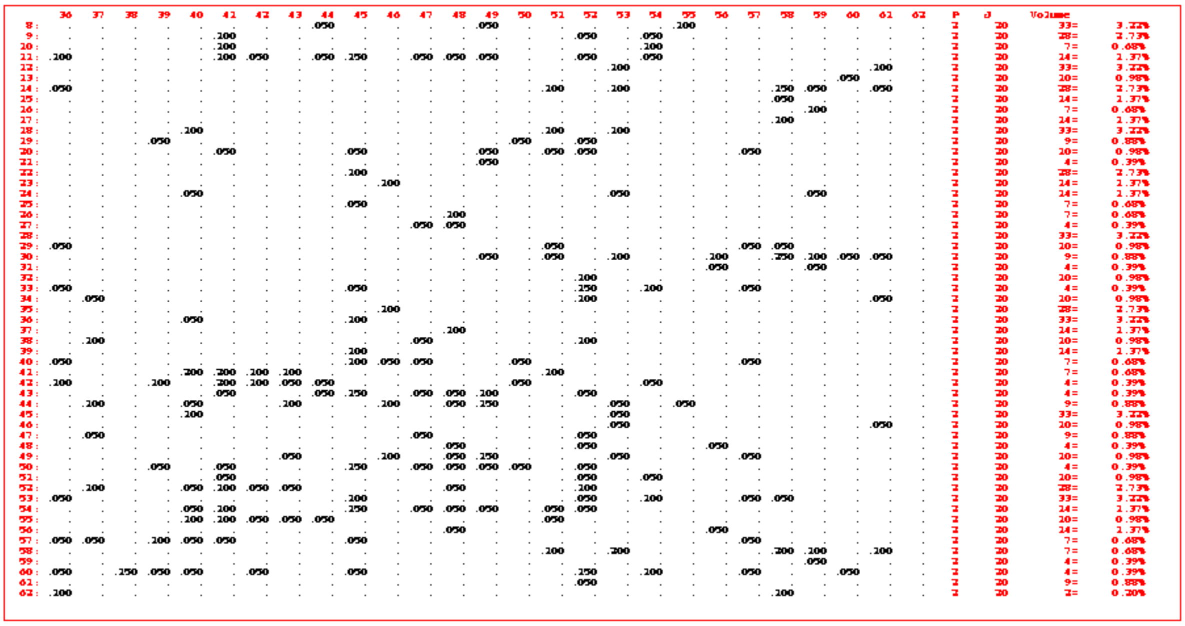


Figure 20.16: Scanning a large jump-table with the options described in section 20.10.1, with 62 basins. The first row has been moved to 8, and the first column to 36, to show the last part of the table. The network is a 1d CA, n=10, k=3, rule 33. Its basin of attraction field and meta-graph is shown in figure 20.17.

When the jump-table or adjacency matrix is visible, the following top right options allow the table to be scanned. This is required for large tables where the data does not all fit inside the window.

scan table: exit-ret jump-j reset-s down-d up-u right-r left-l

The meaning of these prompts is listed below,

exit-ret ... exit the table and return to the meta-graph and prompts in section 20.4.
jump-j ... jump to a new index for the first column and row. The following top right prompt is presented, **jump: enter x: enter y:** Enter the new column and row index.
reset-s ... redraw the default table starting with row 1, column 1.
down-d ... move down, increase the first row index.
up-u ... move up, decrease the first row index.
right-r ... move right, increase the first column index.
left-l ... move left, decrease the first column index.

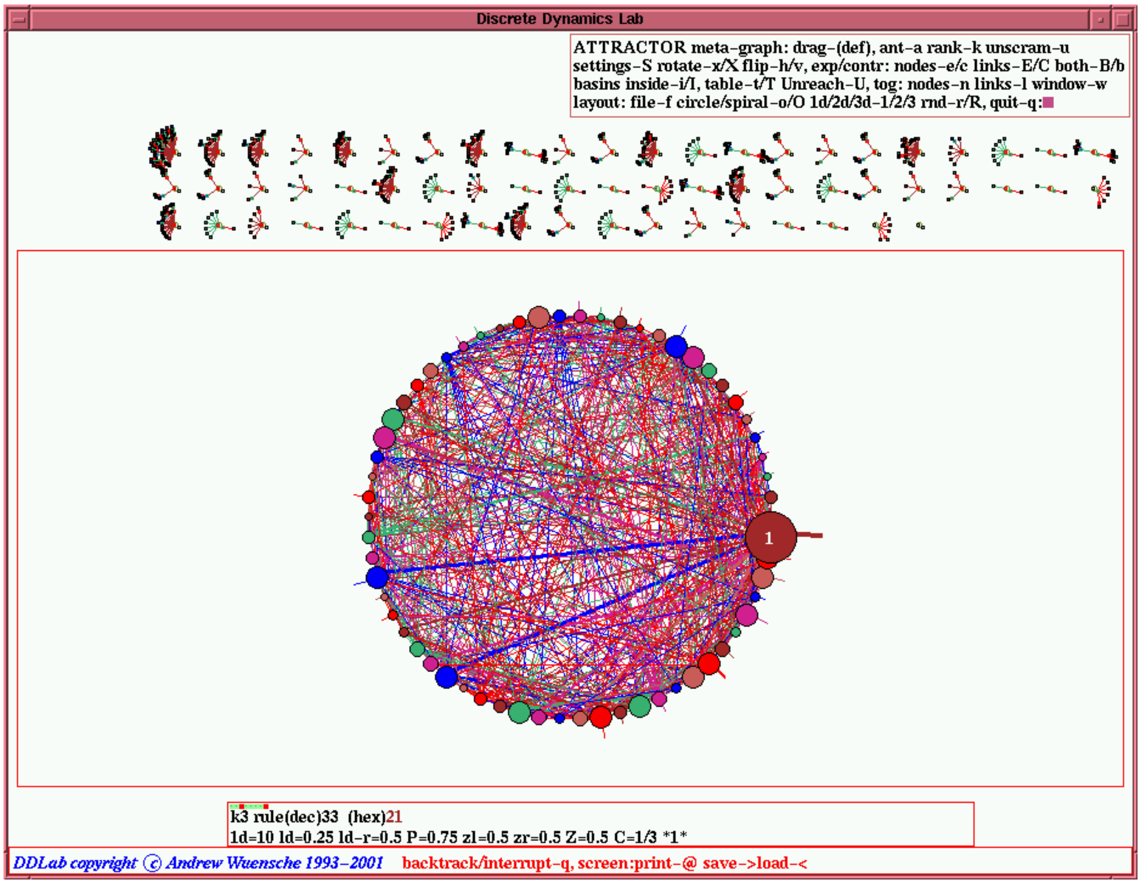


Figure 20.17: The meta-graph of a CA with 62 basins. The network is a 1d CA, n=10, k=3, rule 33. Its jump-table is shown in figure 20.16

Chapter 21

The Seed or initial state

To run a network forwards, or to generate a single basin or subtree (i.e. if **s** was selected in section 6.2), an initial stare or "seed" is required. This chapter describes how the seed may be specified and amended. Various methods of are available. Note that a seed is not required to generate a basin of attraction field.

Section 9.6 described how the network is indexed and arranged in a regular 1d, 2d or 3d array (see figures 9.7 and 9.8). Network states, including the seed, are indexed in the same way, as illustrated in figure 21.1. Specifying a seed will assign a 0 or 1 to each cell in the network array. The methods are similar to "setting a single rule" in chapter 16.

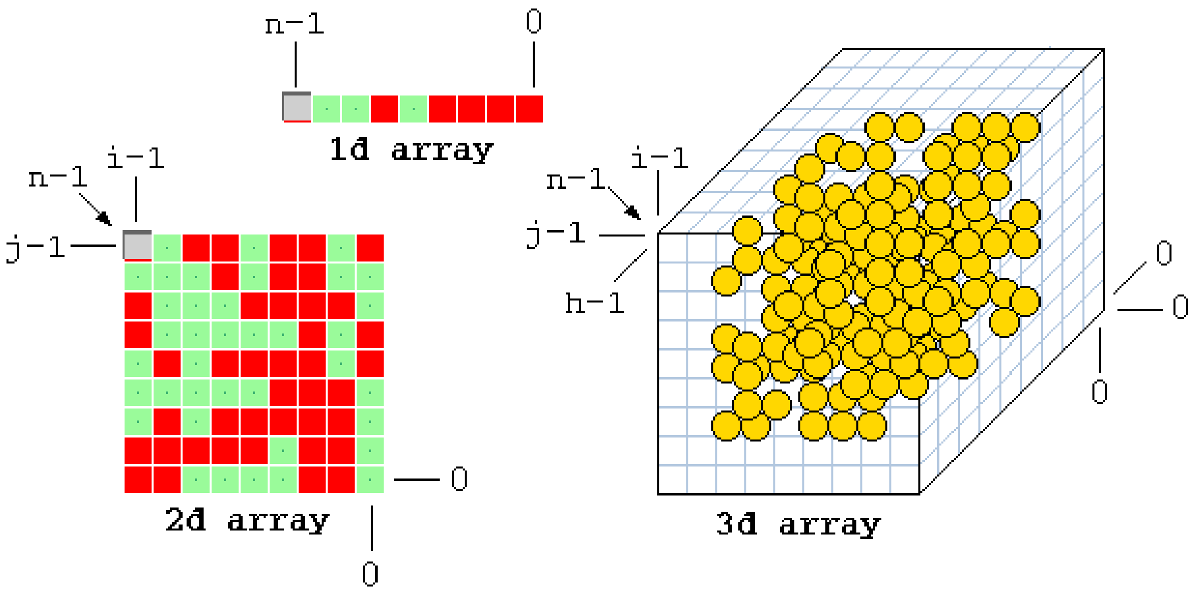


Figure 21.1: The seed (initial state) indexing is the same as network indexing described in section 9.6. The figures show examples for 1d, 2d and 3d seeds and how they are represented. The 3d projection is axonometric seen from below, as if looking up at the inside of a cage.

21.1 Setting the selection method for the seed

Various methods are available for setting the seed. Initially a lower left window is displayed in the main prompt sequence. When reseting a seed while generating space-time patterns or basins, the window appears in the lower right. The size of the window varies according to previous parameters, but can be resized.

The exact wording of the prompt varies according to context (see examples below). In general, when first setting the seed the default is "random", or if a seed was previously set the default is "bits".

example for a 1d network
Select SEED (1d n=14), win-w empty-e fill-f dec-d rand-r bits1d-b bits2d-B hex-h repeat-p load-l (def-r): *(***d** *if $n \leq 16$)*

example for a 2d network
Select SEED (ij=40×40, win-w empty-e fill-f rand-r bits2d-b hex-h repeat-p load-l (def-r):

example for a 3d network
Select SEED (3d ijh=9,9,9, win-w empty-e fill-f rand-r bits2d+3d-b hex-h repeat-p load-l (def-r):

21.2 Methods for setting a seed

A seed can be set in a variety of different ways.

win-w ... allows the seed window to be resized (section 21.2.1).
empty-e ... reset all cells to 0 (section 21.2.2).
fill-f ... reset all cells to 1 (section 21.2.2).
after **w**, **f** *or* **e**, *the program reverts to the prompt in section 21.1.*
dec-d ... in decimal (section 21.7).
rand-r ... at random. (section21.4.
bits-b ... as bits, set in a graphic array using the mouse or keyboard (section 21.6).
bits1d: in a 1d graphic array for 1d networks.
bits2d: in a 2d graphic array for 2d networks.
bits2d+3d: in 2d and 3d graphic arrays shown simultaneously for 3d networks.
bits2d-B ... as bits for a 1d network shown in 2d (1d networks only, section 21.6.4).
hex-h ... in hex, in a mini "spread sheet" (section 21.8).
repeat-p ... a repeat of the last seed that was set (section 21.9).
load-l ... loaded from a `.eed` file (section 21.10).

Enter one of the responses above at the prompt in 21.1 or accept the default.

Once set, the seed remains current and can be amended by resetting bits, in hex, or any other method. The seed can be saved and loaded (texttt.eed file). The size of the seed to be loaded may be different from the base network; a small seed can be loaded at a selected position.

The various methods of setting and amending the seed are described in detail below (they are similar to the rule setting methods in section 16.2).

21.2.1 Resizing the seed window

Sometimes the default seed window may be too small for effectively setting the seed in bits or hex. Enter **w** in section 21.1 to resize the window. The following prompt is presented,

> **change window size (max-m), width (now 480): height (now 550):**
> *(values shown are examples)*

Enter the new window size (minimum 50 $\times$ 50), or enter **m** for the maximum size to fit the screen.

21.2.2 Set seed to all 0's or 1's

Enter **e** or **f** in section 21.1 to "empty" or "fill" the seed. This sets all 0's or all 1's, and allows a clean slate for setting bits (see section 21.6), or amending the seed by some other method.

21.3 Seed bit pattern display

The seed bit pattern is displayed in 1d, 2d or 2d+3d during the selection procedure depending on the dimensions of the network (see figure 21.1).

In 1d and 2d, 0's are colored light grey, 1's are colored red. In 3d, 1s appear as yellow balls suspended in space. Generally a 1d network is displayed in a row, or successive rows for a large array, but can also be shown in 2d. A 2d network is shown as a rectangular or square grid. A 3d network is shown simultaneously in 2d and 3d, where the 2d pattern shows successive horizontal slices of the 3d array. Large 2d arrays like this can be can be panned vertically to focus on different parts. The default scale of the bit pattern graphic varies according to the size of the network, but can be resized.

21.4 Setting the seed at random

If **r** is selected in section 21.1, a seed, or just a a central block, can be set at random. A central block only applies if the length n for 1d, or the width i for 2d, is greater than 20, or if the width i for 3d is greater than 5.

The following prompt is displayed, depending on network dimensions,

> **central segment size (def 30), all-a (max size 148):** *(for a 1d network, n=150)*
> **central 2d block (def 8), all-a (max size 38):** *(for a 2d network, 40×40)*
> **central 3d block (def 6), all-a (max size 18):** *(for a 3d network, 20×20×20)*

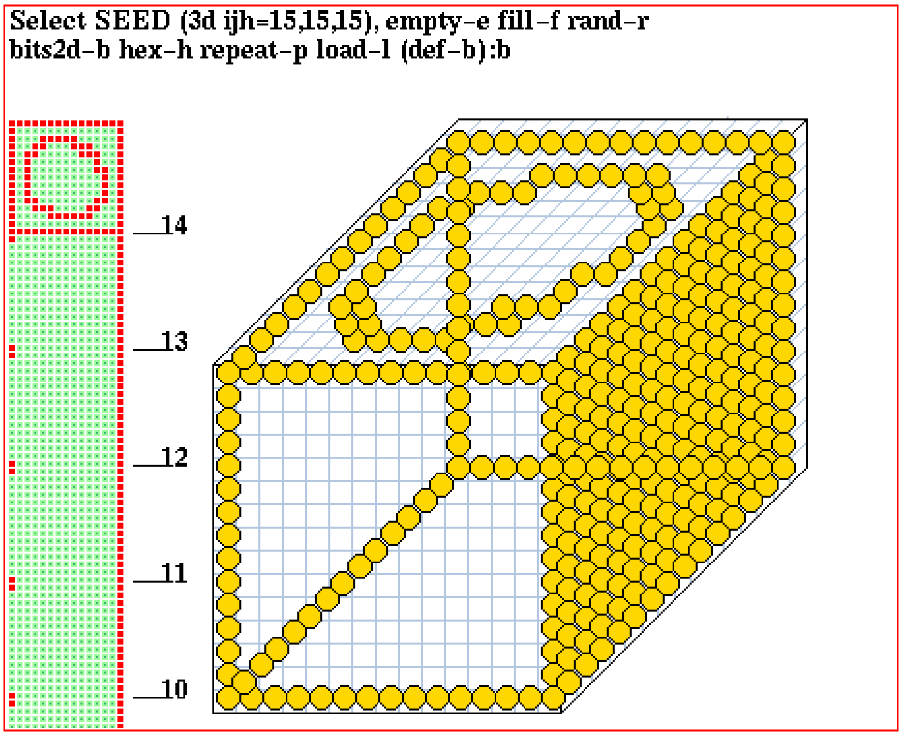


Figure 21.2: An example of the 3d seed window, showing a 3d array, size $15 \times 15 \times 15$, in 2d and 3d simultaneously. Bits drawn on the 2d graphic, are reflected in the 3d graphic. In this example the right side of the 3d array, and all edges, were filled with 1s using the keyboard, and a freehand circle was drawn in the top side with the mouse. Section 21.6 explains the methods. The 2d graphic has the 3d levels indicated, and can be panned vertically to focus on different levels. The 2d and 3d graphics can be independently expanded and contracted.

The default segment size is about 1/5 of the 1d network size, 1/5 of the smallest 2d axis, or 1/3 of the smallest 3d axis. Enter the size of the random block or accept the default, or enter **a** for a random bit pattern across the whole network. The pattern is selected at random and the seed array is displayed as a bit pattern in 1d or 2d. For 3d networks the seed is displayed in 2d only, with 3d levels indicated, as shown in figure 21.2. To show the seed in 3d as well as 2d, enter **q** to revert to the prompt in section 21.1, and select **b** to set the seed as bits.

The seed is also shown in decimal if $n \leq 32$, and in hex if $n \leq 256$. Various changes and transformations to the seed can be made from the random seed window, described below. Similar changes can be made from the "setting bits" window.

21.5 Further random seed options

Further options allow the seed to be changed and transformed from the random seed window. These changes can be made repeatedly until **return** is entered to accept the seed. The following reminder is shown in the random seed window,

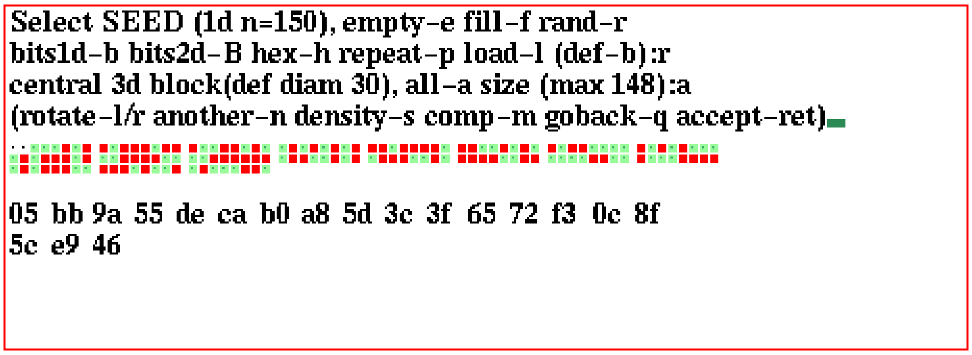


Figure 21.3: The seed selection window, showing a 1d random seed, n=150. The random seed bits are separated into byte size segments, and folded to fit the window. Excess bits are indicated by leading dots. The hex representation of the seed is shown below (for $n \leq 256$).

(rotate-l/r another-n density-s comp-c goback-q accept-ret) *(for 1d)*
(rotate-l/r/u/d another-n density-s comp-c goback-q accept-ret) *(2d and 3d)*

21.5.1 Rotate the random seed

The seed can be "rotated" about the periodic axes in 1d and 2d. In section 21.6, enter **l** and **r** to displace the bit pattern left and right, and for 2d, **u** and **d** for up and down. The bit pattern is displaced within the ring of cells in 1d, and within the torus in 2d.

There are more "rotation" options from the the "setting bits" window in section 21.6, where the rotation interval can be changed, and 3d rotations can be made.

21.5.2 Another random seed

Enter **n** in section 21.5 for another random seed with the same constraints as set in 21.4. That is, the same block size or whole array, with the same density bias (see section 21.5.3).

21.5.3 Bias the density of 1s

The density of 1s in the the random seed or block is displayed in a top right window, together with the current density bias (default 50%). This example is for a 2d network, 40×40,

for the whole network
density 1s=809/1600=50.562%, bias=50.000%
for a central block 30×30
density 1s=435/1600=27.188%, block=435/900=48.333%, bias=50.000%

Enter **s** in section 21.5 to change the density bias, or set the bias to be "strict". The following top right prompt is presented,

enter % (def 50.0%), accept-ret strict-s *(initial density=50%, new setting becomes the default)*

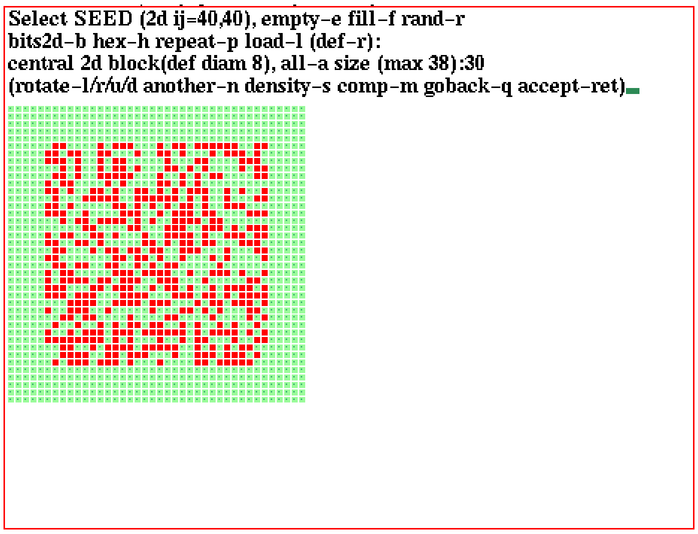


Figure 21.4: The 2d select seed window, showing a random seed consisting of a central random block 30×30 in a 40×40 2d network, with a density bias of 50%.

Enter a new density bias (which becomes the default), then enter **s** for the bias to be applied strictly, so that the new random seed or central block density will be exactly (or very close to) the set bias. Enter **n** (section 21.5.2) to generate the new random seed or block with the new biases.

21.5.4 Compliment the seed

Enter **m** in section 21.5 to compliment the seed, switching 0s and 1s.

21.6 Setting the seed as bits

If **b** or **B** is selected in section 21.1 the seed can be set as bits in 1d, 2d or 2d+3d. 1d networks can be displayed in both 1d (enter **b**) and 2d (enter **B**).

Bits are set with the keyboard or drawn with the mouse on the seed bit pattern display described in section 21.3, and indexed as shown in figure 21.1.

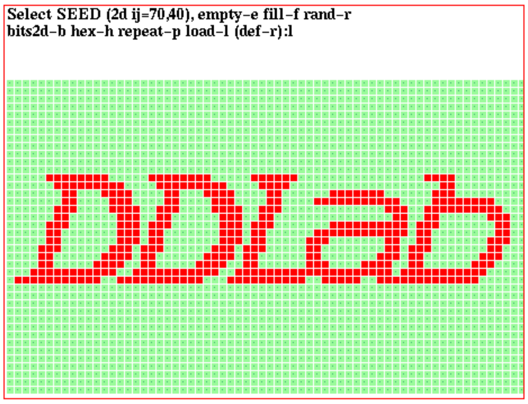


Figure 21.5: Drawing a 2d seed with the mouse or keyboard. Start with a clean slate, all 0s or 1s, or some other pattern. Draw 1s by dragging the mouse cursor over the grid with the left button depressed. Similarly draw 0s with the right button depressed. Horizontal lines can be drawn with the 1 or 0 key, vertical lines with the 2 key (for 1) and the 9 key (for 0). Move without drawing with the arrow keys, or by clicking a new position with the mouse. The drawing is the seed for the DDLab logo.

The default scale of the bit pattern varies according to the dimension and size of the network, but can be rescaled. The current position on the grid is indicated in a small top center window. In UNIX or Linux, if the left mouse button is pressed with the pointer outside the grid, outside grid will appear in this window.

0's are colored grey, 1's are colored red. To provide a clean slate for setting 1's, reset the bit pattern to all 0's in section 21.2.2 with **empty-e**, (or the conversely reset to all 1's with **fill-f** for setting 0's). Alternatively, use this bit setting method to modify seed patterns set by other methods.

Bits in the bit pattern are set to 1 or 0 with the mouse or keyboard. The methods are similar to setting bits for the rule-table described in section 16.10. During the bit setting procedure a top right reminder window displays the following,

example for a 1d network, size 150

keys: set bit h-1/0 v-2/9 move-arrows **1d index=149, rot=1—**
mouse: move-click, draw-keep pressed
buttons: set 1-left, set 0-right
exp/contr-e/c rotate-l/r/+/-, comp-m file-F accept-ret

example for a 2d network, 40×40

keys: set bit h-1/0 v-2/9 move-arrows **2d index ij=39×39, rot=1**
mouse: move-click, draw-keep pressed
buttons: set 1-left, set 0-right, pan2d up/down=U/D exp/contr-e/c
rotate-l/r/u/d+/-, comp-m file-F accept-ret

example for a 3d network

keys: set bit h-1/0 v-2/9 move-arrows **3d index ijh=14×14×14, rot=1**
mouse: move-click, draw-keep pressed
buttons: set 1-left, set 0-right, pan2d up/down=U/D exp/contr-e/c E/C
rotate-l/r/b/f/u/d/+/-, comp-m update3d-t file-F accept-ret

The current bit to be updated is highlighted by a small green flashing cursor (the bit cursor), and its position is displayed in a top right inset window within reminder window above. For example, **3d index ijh=14×14×14, rot=1** indicates that the bit cursor is at position 14×14×14 in a 3d network, and that the "rotation interval" is set to 1 (see section 21.6.8).

To move the cursor, use the left/right/up/down arrow keys relative to the 1d or 2d array. The 1d up/down arrow keys apply to bit patterns that occupy more than one row. For 3d arrays, the bit cursor only appears on the 2d representation of the array. The position of the bit cursor can also be moved by clicking on a new position with the mouse. The array scale can be expanded and contracted.

Bits are set from the keyboard (section 21.6.1) or drawn with the mouse (section 21.6.2). The bit pattern can be "rotated" (sections 21.5.1, 21.6.7 and 21.6.8) and complimented (sections 21.5.4 and 21.6.9). The seed can be saved to a `.eed` file, and other seeds can be loaded, including smaller seeds at specified positions.

To accept the bit pattern, enter "return". When selected, the decimal (for $n \leq 32$) and hex (for $n \leq 256$) equivalents of the bit pattern are displayed.

21.6.1 Seed: Setting bits from the keyboard

Press keys **1** and **0** (without **return**) to set bits to 1 or 0. This automatically moves the cursor one position right, so horizontal lines of 1s and 0s can be drawn by holding down the keys. Similarly, vertical lines of 1s and 0s can also be drawn downwards, key **2** draws 1s and key **9** draws 0s.

21.6.2 Seed: Setting bits with the mouse

While the cursor is within the bit pattern grid, the mouse can be used to draw 1's and 0's. Dragging the mouse with the left button depressed draws 1's (i.e. sets bits), the right button draws 0's (i.e. deletes bits).

There is a slight difference between DOS and Unix/Linux. In DOS the mouse cursor is confined within the bit pattern grid and just clicking the left or right mouse button sets single 1s or 0s. In Unix the cursor is not confined, but will change direction within the grid, pointing north-west instead of north-east. Within the grid, left or right mouse clicks reposition the small green flashing cursor for setting bits from the keyboard, but do not set 1s or 0s, the mouse needs to be dragged to do this. The center button in Unix/Linux does not play a role.

21.6.3 Setting bits for 1d segments

For a 1d network, if **b** is selected at 21.1, the following prompts allow just a segment of the seed to be defined,

enter segment to define as bitstring, max/def 200: *(for n=200)*

Once the size of the segment is specified, a further prompt positions the segment within the network,

enter start position from the left 0-149, def centred: *(for a segment size=200)*

Enter the start index of the segment. Note the above does not apply to 2d or 3d networks. A bit pattern representing the current seed (or segment) is displayed for setting bits.

21.6.4 Setting bits for 1d networks shown as 2d

For a 1d network, if **B** is selected at 21.1, the 1d network is represented in 2d as a square or rectangular bit pattern, $i \times j$. Default j (the number of rows) is set automatically as the highest factor of n, $\leq \sqrt{n}$. The following prompt allows the default i to be revised,

now 15 x 14, reset i-axis: *(for n=200)*

The 1d network is broken up into successive rows starting with the maximum cell index in the top left hand corner. If i is not a factor of n, some cells will be missing from the bottom row. If this option is set, the default node display for attractor basins will be set identically (see 25.2.2 and 26.2).

21.6.5 Panning 2d arrays up and down

Enter **U** or **D** in section 21.6 to pan a 2d array up and down. A 2d network will pan by 1 row. A 2d array representing a 3d network (see figure 21.2) will pan by 1 level. This allows different parts of large 3d networks to be brought into focus for setting bits.

21.6.6 Expanding and contracting the scale

Enter **e** or **c** in section 21.6 to expand and contract the scale of a 1d or 2d bit array, and **E** or **C** to expand and contract the 3d bit array. Note that the 2d and 3d representations a 3d array, shown simultaneously as in figure 21.2, can be rescaled independently.

21.6.7 Setting bits: rotating

The seed can be "rotated" about the periodic axes in 1d, 2d and 3d. In section 21.6, enter **l** or **r** to displace the bit cursor left or right, **u** or **d** for up or down, and **f** or **d** for forwards or backwards in a 3d volume. The bit pattern is displaced within the ring of cells in 1d, within the torus in 2d, and within the 3-torus for 3d.

21.6.8 Setting bits: rotation interval

Enter + and **-** in section 21.6 to increase or decrease the rotation interval, the number of cell by which the seed will be rotated in 1d, 2d or 3d, the default=1 (see section 21.6.7). The current rotation interval is displayed in a top left inset within the reminder window in section 21.6. For example **3d index ijh=14×14×14, rot=3** indicates that the bit cursor is at position 14×14×14 in a 3d network, and that the rotation interval is set to 3.

21.6.9 Setting bits: Complimenting the seed

Enter **m** in section 21.6 to transform the bit pattern into its compliment, changing 1s to 0s and vice versa.

21.6.10 Setting bits: Updating 3d

For 3d networks, after drawing bits with the keyboard or mouse on the 2d representation of the array (sections 21.6.1 and 21.6.2), enter **t** in section 21.6 to update the 3d array.

21.6.11 Setting bits: filing

Enter **F** in section 21.6 to load as many seed files as required into the current seed, or to save the seed. The following top right prompt is displayed,

SEED: load-l save-s:

Enter **l** to load a seed, which will be immediately displayed, and may be smaller or bigger than the current network. If smaller, the seed's position within the network can be specified. Enter **s** to save the seed.

For further details on loading see section 21.10, on saving see section 21.11, and for the encoding of the `.eed` seed file, see section 21.11.1.

21.7 Setting the seed in decimal

For network size, $n \leq 31$, if **d** is selected in section 21.1, the seed can be specified by its decimal equivalent. The following prompt is presented,

decimal seed, enter 0-16363 (def-rand): *(example for n=14)*

If just **return** is entered at this prompt, or a number outside the permitted range, the selection will revert to "random" (section 21.4).

21.8 Setting the seed in hex

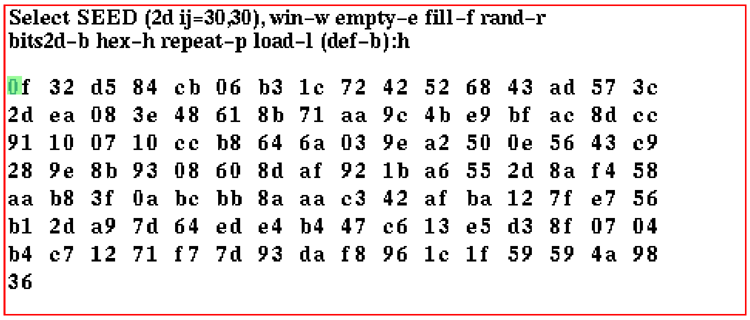
If **h** is selected in section 21.1, the seed may be set in hex. The hex expression of the current seed is displayed. The hex character at the green flashing cursor can be overwritten from the keyboard (enter the hex numerals **0-9** or **a-f**). The cursor will automatically move to the next digit. The left/right (and up/down) arrow keys are use to displace the cursor. The up/down arrow keys apply if there are more than 2 rows of hex.

The following top right reminder is displayed,

enter hex, arrows-move
rotate-l/r *(for 2d or 3d* **-l/r/u/d***)*

The seed can be "rotated" about the periodic axes in 1d and 2d as in section 21.5.1. Enter **l** and **r** to displace the bit pattern left and right, and for 2d, **u** and **d** for up and down. The hex expression changes accordingly.

To accept the hex seed enter **return**, the resulting seed bit pattern will be displayed in 1d, 2d or 2d+3d, in bit setting mode described in section 21.6. Enter **return** to accept.


```
Select SEED (2d ij=30,30), win-w empty-e fill-f rand-r
bits2d-b hex-h repeat-p load-l (def-b):h

0f 32 d5 84 cb 06 b3 1c 72 42 52 68 43 ad 57 3c
2d ea 08 3e 48 61 8b 71 aa 9c 4b e9 bf ac 8d cc
91 10 07 10 cc b8 64 6a 03 9e a2 50 0e 56 43 c9
28 9e 8b 93 08 60 8d af 92 1b a6 55 2d 8a f4 58
aa b8 3f 0a bc bb 8a aa c3 42 af ba 12 7f e7 56
b1 2d a9 7d 64 ed e4 b4 47 c6 13 e5 d3 8f 07 04
b4 c7 12 71 f7 7d 93 da f8 96 1c 1f 59 59 4a 98
36
```

Figure 21.6: Setting the seed in hex. In this example for a 2d network, 30×30, the seed was first set at random, then reselected in hex. The hex character at the green flashing cursor can be overwritten from the keyboard (enter the hex numerals **0-9** or **a-f**). The cursor will automatically move to the next digit. The left/right/up/down arrow keys displace the cursor.

21.9 Repeating the seed

Enter **p** in section 21.1, to repeat the current seed.

21.10 Loading a seed

Enter **l** in section 21.1, or **F** followed by **l** in section 21.6.11 to load a seed from a `.eed` file, the default filename is `myseed.eed`. A top right windows allow a seed be loaded (see Filing, chapter 35).

21.10.1 Loading seeds smaller than the base network

A file seed that is smaller than the base network can be loaded in a set position. If the file is bigger, as many bits as will fit are loaded (see section 21.10.2). Preferably the dimensions (1d, 2d or 3d) of the file and base network should be the same, but if not, the file can still be loaded. The following rules apply,

1. For a 1d file, size n, to be loaded into a 2d or 3d base network, width i, provided $n \leq i$, the 1d file will be converted to a 2d or 3d file.
2. For a 2d file, $i_2 \times j_2$, to be loaded into a 3d base network, $i \times j \times h$, as long as $i_2 \leq i$ and $j_2 \leq j$, the 2d file will be converted to a 3d file.
3. Otherwise, if the dimensions do not correspond, the file will be loaded as if both file and base were 1d networks.

Once the file has been selected (see Filing, chapter 35), if the file seed is smaller than the base network in all dimensions, one of the following prompts is presented, depending on the dimensions, to position the smaller file somewhere on the larger base array. The default is centrally. The existing pattern in the base network outside the area where a smaller seed is loaded will not be effected.

example for a 1d network
1d: array length=100, file length=20, enter start pos
(def 40, max 80): *(values shown are examples)*

example for a 2d network
2d:i,j=100,100, file:i,j=6,6, enter start coords
(def 47,47, max 94,94) i: j: *(values shown are examples)*

example for a 3d network
3d:i,j,h=40,40,40, file:i,j,h=15,15,15, enter start coords
(def 12,12,12 max 25,25,25) i: j: h: *(values shown are examples)*

Enter the start coordinates for positioning index 0 of the file seed, or enter **return** for the default central position. Any number of file seeds can be loaded.

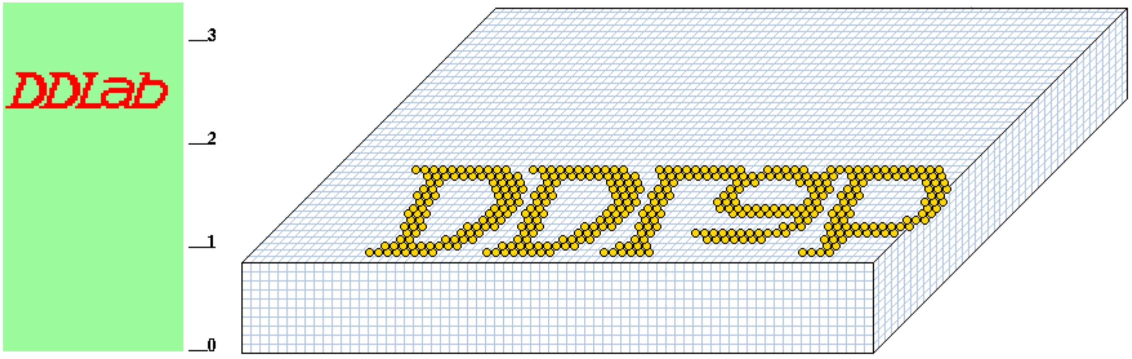


Figure 21.7: Loading a 2d seed into a 3d network, 70 × 40 × 10. The 2d seed is the DDLab logo shown in figure 21.6, loaded into level 2 of the 3d network. The 3d projection of the network is shown, and part of the 2d representation on the left.

21.10.2 Loading seeds bigger than the base network

The following rules apply if a file seed that is bigger than the base network is loaded,

1. For 1d (file and base), as many bits in the file seed as will fit into the base seed will be loaded starting at index 0.
2. For 2d and 3d (file and base), if each dimension of the base is smaller or equal to each dimensions of the file, the central portion of the file seed will be assigned to the base network.
3. If the requirements in 2 are not met, or if the dimensions of the file and base do not correspond, as many bits as will fit will be loaded starting at index 0, as for 1d.

21.11 Saving a seed

Enter **F** followed by **s** in section 21.6.11 to save the seed. Alternatively once the seed has been accepted, a top right window shows the density of 1s (see section 21.5.3) and allows the seed to be saved or revised,

> **density 1s=341/729=46.776%, bias=50.000%**
> **SEED: revise-q save-s cont-ret** *(values shown are examples)*

Enter **s** to save the seed (as a `.eed` file). A top right filing window will appear (see Filing, chapter 35) allowing the seed to be saved (the default filename is `myseed.eed`.

21.11.1 Seed file encoding

The `.eed` file defining the seed is encoded as follows, where n is the network size, (i, j) the axes of a 2d network, and (i, j, h) the axes of a 3d network,

byte 0,1 ... network size, n.
byte 2,3 ... i, j.
if $i = 0$ and $j = 0$ the seed is 1d.
if both $i > 1$ and $j > 1$ the seed can be either 2d or 3d.
$h = n/(i \times j)$. If $h > 1$ the seed is 3d, otherwise 2d.

The rest of the seed is set as bits from 0 to $n - 1$ in successive bytes. Excess bits in the last byte are set to 0.

Chapter 22

The Derrida plot

If **D** is selected from the wiring graphic in chapter 17, (sections 17.4.2 or 17.5), the Derrida plot (and Derrida coefficient) will be generated. Any type of network may be plotted, a CA or RBN, in 1d, 2d, or 3d. The main application of the Derrida plot is as an order-chaos measure for large RBN networks in the context of models of genetic regulatory networks[4, 7], where the canalizing inputs can be tuned to move the dynamics between order and chaos (see section 15). The Derrida plot provides a statistical measure of the divergence/convergence of network dynamics in terms of "Hamming distance" between states. The Hamming distance between two binary states of equal size, n, is the number of bits that differ, H. The normalized Hamming distance is H/n.

The Derrida plot is generated as follows,

1. Randomly select a pair of initial states, I_1, I_2, separated by a small Hamming distance of H_0, at time-step t_0.
2. Iterate each state forward independently, according to the network architecture, by usually one, or more time-steps, i. i must be 1 to compute the Derrida coefficient.
3. Measure the Hamming distance, H_i, at time-step i, between the pair of final states, F_1, F_2.
4. Repeat the above for a sample of pairs of initial states with the same initial Hamming distance H_0, and plot normalized H_0 (x-axis) against the mean normalized value of the H_i's, (y-axis).
5. Repeat the whole procedure for a range of increasing initial Hamming distances. The increase (Ham steps) must be 1 to compute the Derrida coefficient, but may be set to larger intervals otherwise.

22.1 The Derrida coefficient

A measure drived from the Derrida plot is the Derrida coefficient, analogous to the Liapunov exponent in continuous dynamical systems. The initial slope of the Derrida plot, a tangent to the curve at the origin, indicates whether the network dynamics is convergent or divergent, ordered or chaotic, and to what extent.

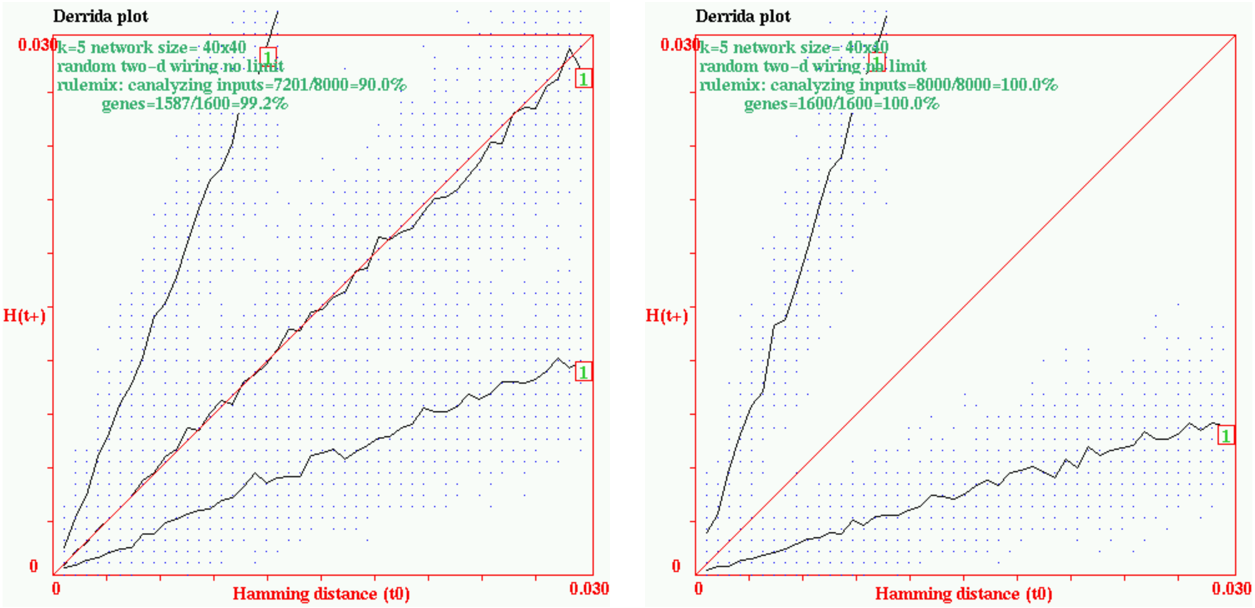


Figure 22.1: The Derrida coefficient taken from the initial slope of the Derrida plot, for a 2d random Boolean network, $k = 5$, $n = 40 \times 40$. Data about the network are shown in the top left hand corner of the graph (see section 22.4). Parameters were as follows: Max$H$=0.03, sample=55, iterations=1, Ham steps=1. The Derrida coefficient sample was set to 15. If the avarage slope of the first 15 points $= \delta$ degrees, then the Derrida coefficient, $Dc = log_2(tan(\delta))$.
left: 3 superimposed plots for Canalizing settings $C = 0$, 51%, 90%, are shown, with progressivly shallower slopes. The the Derrida coefficient measures given in the top right window were,
C=0: **slope=68.0 degrees Dc=1.305**
C=51%: **slope=45.1 degrees Dc=0.007**
C=90%: **slope=23.2 degrees Dc=-1.220**
right: 2 superimposed plots, pushing the extreems of order/chaos behaviour to their limits. The steep slope was generated by an RBN with maximally chaotic chain rules (see sections 16.7 and 13.4),
the measures were: **slope=71.5 degrees Dc=1.577**.
The shallow slope was generated by a maximally ordered RBN with 100% canalizing inputs,
the measures were: **slope=17.1 degrees Dc=-1.701**.

A curve with an initial slope above the $x = y$ diagonal (i.e. above 45 degrees) indicates chaos, below indicates order, a slope close to 45 degrees indicates balanced dynamics.

The initial slope is based on the first H points, for one time-step iterations (i=1) and a range of Hamming steps increasing by 1. The number of points (the Dc sample) is selected to lie approximately on a straight line. The default is 15 for large networks. The average slope between these points and the origin is taken as the initial slope. If the initial slope $= \delta$ degrees, then the Derrida coefficient, $Dc = log_2(tan(\delta))$.

A slope of 45 degrees ($Dc = 0$) indicates the dynamics is ballanced between order and chaos. A slope above the 45 degree diagonal (positive Dc) is in the chaotic regime, below (negative Dc) is in the ordered regime. The degree of chaos/order is given by the the slope and Dc. Figure 22.1 *left* gives examples for a *k=5* RBN, where a random (chaotic) rule mix (no canalizing inputs) has a slope of 68.0 degrees ($Dc = 1.305$) and a highty ordered rule mix (90% canalizing inputs) has a slope of 23.2 degrees ($Dc = -1.220$). An RBN were the canalizing inputs were tuned to 51% gives a slope of 45.1 degrees ($Dc = 0.007$)

Pushing the extremes of order/chaos behaviour to their limits, figure 22.1 *right* shows 2 plots, an RBN with maximally chaotic chain rules (see sections 16.7 and 13.4) which gives a slope of 71.5 degrees ($Dc = 1.557$), and a maximally ordered RBN (canalizing inputs=100%) which gives a slope of 17.1 degrees ($Dc = -1.701$).

22.2 Selecting the Derrida plot

To generate the Derrida plot, first display the network as a wiring graphic (section 17.1. Then from among the top right wiring graphic prompts (sections 17.4.2, 17.5.2, or 17.6.2, for 1d, 2d or 3d networks), the following prompt is presented,

... Derrida plot-D:

Enter **D** to select the Derrida plot.

22.3 The Derrida plot parameters

If **D** is selected in section 22 above, the following series of prompts are presented in sequence in a top right window. This allows the default Derrida plot parameters to be changed. The revised parameters generally become the new defaults,

Derrida plot: quit-q, De sample (now 15, max 320): *(for a* 40×40 *RBN)*
followed by parameter prompts as follows (values shown are examples)
maxH (now 1.0): sample (now 25) suppress fuzz-c: show details-s
iterations (now 1): Ham steps (now=1 net=1600): keep-k:

The various parameters are summarized below, and described in more detail in sections 22.3.1 - 22.3.8.

	key to Derrida plot parameters
quit-q ...	quit the Derrida plot.
De sample ...	for the Derrida coefficient, the number of initial points to be included to calculate the initial slope.
maxH ...	the maximum normalized Hamming distance in the range.
sample ...	the sample size for each initial Hamming distance.
suppress fuzz ...	suppress plotting the spread of each sample.
details ...	for each sample point, pause to show H_0 and H_i.
iterations ...	the number of forward time-steps.
Ham steps ...	the interval between increasing initial Hamming distances.
keep ...	keep the previous plot and overlay the new plot.

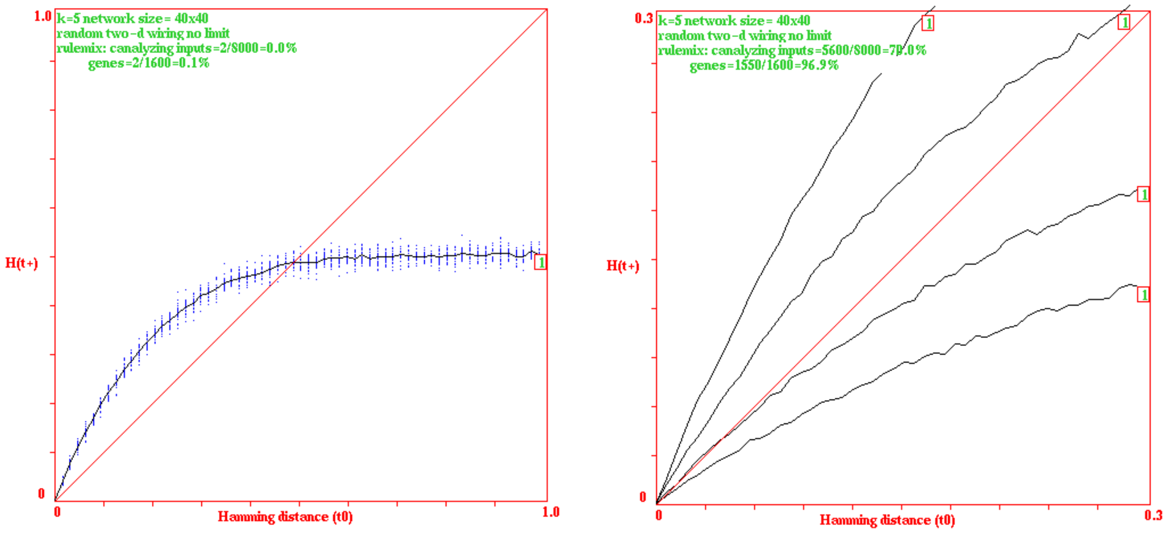


Figure 22.2: Examples of Derrida plots for a 2d random Boolean network, $k = 5$, $n = 40 \times 40$. Data about the network are shown in the top left hand corner of the graph (see section 22.4). *left:* Max$H$=1, sample=25 (displayed), iterations=1 (indicated at the end of each curve), Ham steps=25. *right:* 4 superimposed plots for increasing Canalizing settings (0, 2%, 50% and 70%), with progressivly lower slopes. Max$H$=0.3, sample=25 (suppressed), iterations=1, Ham steps=10.

22.3.1 The Derrida coefficient sample

At the prompt **De sample (now 15, max 320):** (for example) enter the number of inital points that will form the basis of the Derrida coefficient, derived from the initial slope (see section 22.1). The initial default depends on the size of the network. Generaly, between 10 and 20 points are appropriate.

22.3.2 Maximum Hamming distance

At the prompt **maxH:** in section 22.3, enter a number, $0 < maxH \leq 1$, to set the maximum normalized Hamming distance of the plot. The initial default is 1. A small $maxH$ shows a detail of just the left hand corner of the plot, which is perhaps the most interesting. Only a small initial part of the plot is required to calculate the Derrida coefficient, enough to contain the Derrida coefficient sample described in 22.3.1 above.

22.3.3 Sample size

At the prompt **sample:** in section 22.3, enter the sample size for each initial Hamming distance. The initial default is 25. A smaller sample gives a quicker plot. A larger sample provides a more accurate measure.

22.3.4 Suppress the sample spread plot

Enter **c** at the prompt **suppress fuzz:** in section 22.3, to suppress plotting each sample point, showing the spread of data. Plotting the sample data is the default. Note that the plot can be be redrawn from memory without the sample spread "fuzz" (see section 22.6.1).

22.3.5 Derrida plot details

Enter **s** at the prompt **details:** in section 22.3, to pause at each sample point and show detailed data in a top right window (mainly for debugging and diagnostic purposes), for example,

sample=4 H(t0)=21 H(t1)=16 q-quit: *(values shown are examples)*

In this example, at the 4th sample point, the initial Hamming distance is $H_{t0} = 21$ (not normalized), the final Hamming distance, is H_{t1}=16, at time-step 1. Enter **q** to disable the pause and continue without interruption.

22.3.6 Set iterations

At the prompt **iterations:** in section 22.3 enter the number of forward time-steps required. The default is 1. This must be set to 1 to compute the Derrida coefficient.

22.3.7 Set interval

At the prompt **Ham steps:** in section 22.3 enter the size of the interval between successive initial Hamming distances (not normalised). The minimum and default is 1. A small interval provides a more accurate plot. A larger interval provides a quick and sketchy plot. This must be set to 1 to compute the Derrida coefficient.

22.3.8 Keep the previous plot

Enter **k** at the prompt **keep:** in section 22.3, to keep a previous plot and superimpose a new plot, which may have revised network parameters.

22.4 Network data within the Derrida plot

The graph will be labeled with network data in the top left hand corner as follows,

	key to network data within the Derrida plot
k=k ...	for homogenious k.
k-mix=k_1**-**k_2 ...	the range of k for a k-mix .
network size=n ...	for a 1d network.
network size=$i \times j$ ...	for a 2d network.
network size=$i \times j \times h$ ...	for a 3d network.
one-d CA wiring ...	for regular 1d CA wiring.
two-d CA wiring ...	for regular 2d CA wiring.
three-d CA wiring ...	for regular 3d CA wiring.

random one-d wiring ...	for random 1d wiring.
random two-d wiring ...	for random 2d wiring.
random three-d wiring ...	for random 3d wiring.
no limit ...	random wiring is not restricted.
zone=r ...	random wiring is restricted within a radius r.
rulemix: canalizing ...	... for a rulemix,
inputs=c/c_{max}**=**c**%** ...	the percentage of canalizing inputs.
genes=g/n**=**g**%** ...	the percentage of canalizing genes.
homogeneous CA rule ...	for a network with a single rule.

22.5 Interrupting the Derrida plot

While the Derrida plot is in progress, the following top right message is displayed,

plotting Derrida, sample size=25, iteration limit=1
Hamming step=25, quit/pause-q *(values shown are examples)*

If the plot is interrupted with **q**, the following prompts are presented in succession in the top right window,

interrupted at Hamm=0.297, quit now-q, cont-ret:
if **return**, *the parameter prompts as in section 22.3 (values shown are examples)*
maxH (now 1.0): sample (now 25) suppress fuzz-c: show details-s
iterations (now 1): Ham steps (now=1 net=1600): keep-k:

Enter **q** to stop the plot at the point reached and continue with section 22.6 below. Any of the parameters described in section 22.3 may be reset and the plot continued.

22.6 Completing the Derrida plot

When the Derrida plot is complete (or if **return** is entered after it is interrupted in section 22.5), a reminder of the number of iterations is displayed at the end of the curve, and the following Derrida coefficient data and further prompts are displayed in a top right window as follows,

Derrida plot complete, slope=67.7 Dc-1.288 (15 steps)
sample size=25, reset-r canalizing-C: *(values shown are examples)*

Note that the Derrida coefficient data is only given if both the number of iterations (section 22.3.6 and the interval (section 22.3.7) were set to 1. **slope=67.7** is the initial slope of the Derrida plot and **Dc** is the derrida coefficient described in section 22.1.

Further prompts are described below.

22.6.1 Resetting the Derrida plot

Enter **r** to reset the plot as described in 22.3 above, the following successive prompts are presented in the top right window,

> **Derrida plot: redraw (no fuzz)-r, quit-q, Dc sample (now 15, max 320):**
> *followed by the parameter prompts as in section 22.3 (values shown are examples)*
> **maxH (now 1.0): sample (now 25) suppress fuzz-c: show details-s**
> **iterations (now 1): Ham steps (now=1 net=1600): keep-k:**

Enter **r** for a quick redraw of the plot from memory without the sample spread "fuzz". Enter **return** to amend any of the parameters described in section 22.3 and regenerate the plot.

22.6.2 Resetting canalizing for the Derrida plot

Enter **C** in section 22.6 to revise the network's canalising settings. For a mixed rule network see section 15. For a single rule network, see section 18.7.

The Derrida plot can then be regenerated for the revised network. If required, previous plots can be retained (see section 22.3.8) to produce a multiple plot as in figure 22.2(*right*).

Chapter 23

Graphic conventions for attractor basins

Figure 23.1 explains the idea of basins of attraction in discrete dynamical networks. Attractor basins are represented by state transition graphs, where nodes - network states, are linked by directed arcs - state transitions. States in deterministic networks have one successor but possibly a number of predecessors (pre-images), so trajectories occur within trees. In a finite network a trajectory must encounter a repeat state, defining its attractor. Attractor basins are thus made up of transient trees rooted on attractor cycles.

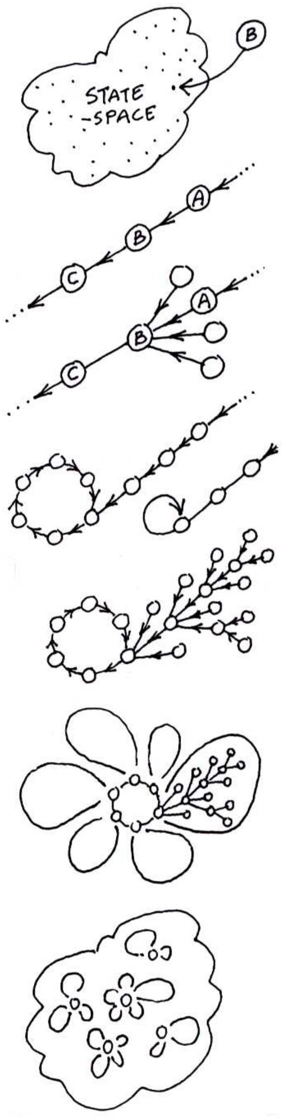
The attractor cycle may have an arbitrary period, including a period of just one, a point attractor cycling to itself. Transient trees may consist of just one node outside the attractor, or a number of nodes with arbitrary in-degree (but one out degree) at each node. Nodes with zero in-degree, the leaves of the trees, are the so called garden-of-Eden states. The graphic conventions for drawing attractor basins and subtrees are described below, and illustrated in figure 23.2 - 23.5 and in other figures in the manual.

23.1 Network states, nodes

Network states in attractor basins are usually represented by circular nodes, but may also be shown as a bitstring pattern in 1d or 2d, as the decimal or hex value of the state, or they may not be shown (see section 26.2).

23.2 Attractor cycles

Where 3 or more states make up an attractor cycle, this is represented as a regular polygon with nodes at the vertices (figure 23.5). The direction of time is clockwise. The “last” vertex, from which the first subtree is generated, is shown due east, so the first vertex at which the cycle is entered is one node clockwise from due east. A point attractor is shown as a node (positioned north-east) on a circle, indicating that the node cycles to itself (figure 23.2). A two state attractor is shown as two nodes on a circle (positioned south-east and north-west), where the south-east node is the “last” node. The diameter of attractor cycles increases asymptotically with system size n, up to the maximum selected, which also determines the scale of the entire basin.



For a network size n, an example of one of its states B might be $1010\ldots0110$. *State-space* is made up of all 2^n states, the space of all possible bitstrings or patterns.

Part of a *trajectory* in state-space, where C is a successor of B, and A is a *pre-image* of B, according to the dynamics of the network.

The state B may have other pre-images besides A, the total number is the *in-degree*. The pre-image states may have their own pre-images or none. States without pre-images are known as *garden-of-Eden* states.

Any trajectory must sooner or later encounter a state that occurred previously - it has entered an attractor cycle. The trajectory leading to the attractor is a *transient*. The period of the attractor is the number of states in its cycle, which may be just one - a point attractor.

Take a state on the attractor, find its pre-images (excluding the pre-image on the attractor). Now find the pre-images of each pre-image, and so on, until all garden-of-Eden states are reached. The graph of linked states is a *transient tree* rooted on the attractor state. Part of the transient tree is a subtree defined by its root.

Construct each transient tree (if any) from each attractor state. The complete graph is the *basin of attraction*. Some basins of attraction have no transient trees, just the bare "attractor".

Now find every attractor cycle in state-space and construct its basin of attraction. This is the *basin of attraction field* containing all 2^n states in state-space, but now linked according to the dynamics of the network. Each discrete dynamical network imposes a particular basin of attraction field on state-space.

Figure 23.1: This figure explains the idea of basins of attraction in discrete dynamical networks. Given an invariant network architecture and the absence of noise, the dynamics is deterministic, and follows a unique trajectory from any initial state. When a state that occurred previously is re-visited, which must happen in a finite state-space, the dynamics become trapped in a perpetual cycle of repetitions defining the attractor (state cycle) and its period (minimum one, a stable point). The approach is analogous to Poincaré's "phase portrait" in continuous dynamics. These systems are dissipative. A state may have multiple "pre-images" (predecessors), or none, but just one successor. The number of pre-images is the state's "in-degree". In-degrees greater than one require that transient states exist outside the attractor. Tracing connections backwards to successive pre-images of transient states reveals a tree-like topology where the "leaves" are states without pre-images, known as garden-of-Eden states. Conversely, the flow in state-space is convergent. The set of transient trees and the attractor on which they are rooted make up the basin of attraction. *Local* dynamics connects state-space into a number of basins, the basin of attraction field, representing the system's *global* dynamics.

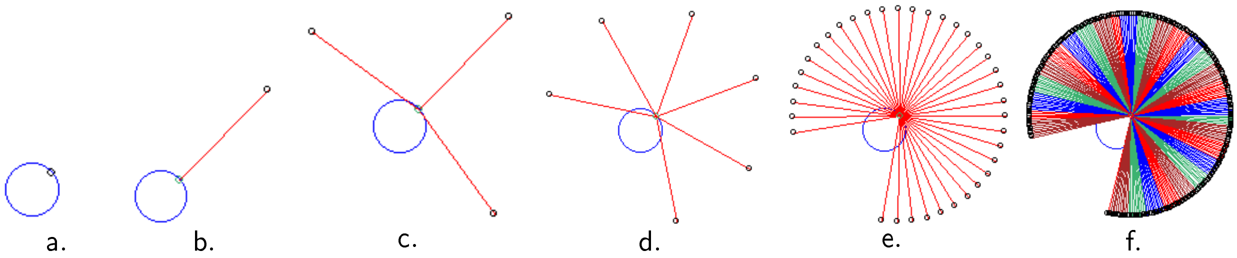


Figure 23.2: A point attractor (period=1) is represented as a node cycling to itself. The center line of the pre-image fan (level 1) is set at a default angle of 45 degrees. The examples above (a - f) show point attractors with increasing fan size, from 0 upwards. Figure (f) shows equivalent pre-images as described in section 23.3.2.

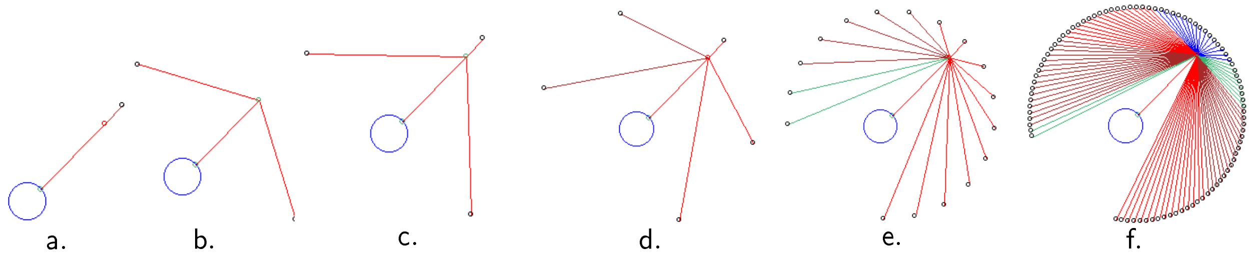


Figure 23.3: The pre-image fan (level 2) of a single pre-image (level 1) of a point attractor. The examples above (a - f) show increasing fan sizes, from 0 upwards. Figure (f) shows equivalent pre-images as described in section 23.3.2.

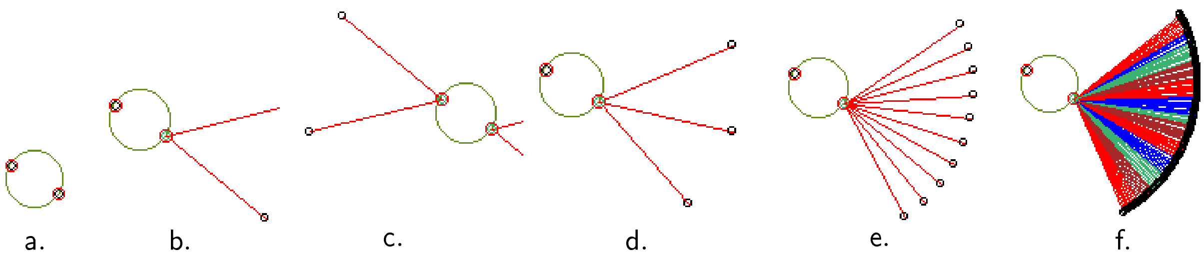


Figure 23.4: Examples of attractors with period 2 and varying sizes of the pre-image fan, from 0 upwards, are shown above. A fan at level 1 is slightly angled to indicate that the direction of time is clockwise in the attractor. The center lines of subsequent fans are radial to the center of the attractor cycle. Figure (f) shows equivalent pre-images as described in 23.3.2.

23.3 Transient trees

The pre-images or predecessors (if any) of the "last" attractor state are computed first. The pre-image fan angle is set according to the in-degree and increases asymptotically to a maximum value with increasing in-degree. In general the cent-re line of the fan angle is radial to the attractor cycle

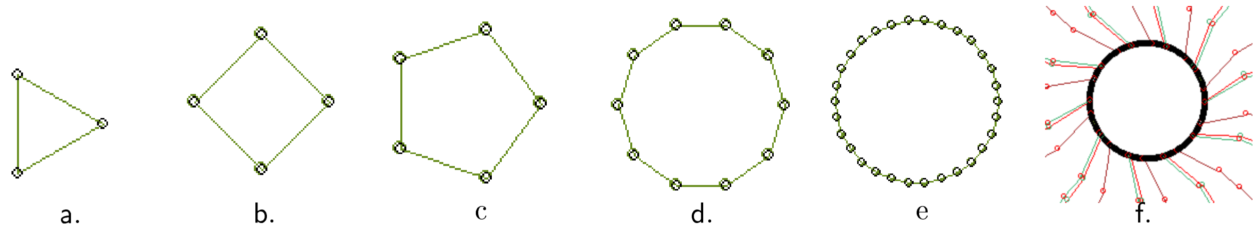


Figure 23.5: Attractors with period 3 and above are represented by polygons. The diameter of the polygon approaches an upper limit with increasing period. The examples above have periods of 3, 4, 5, 10, 30, and 430. A fan at level 1 is slightly angled to indicate that the direction of time is clockwise in the attractor. The center lines of subsequent fans are radial to the center of the attractor cycle.

cent-re point, but for fans rooted on the attractor, the fan angle is tilted slightly to indicate that the direction of time around the cycle is clockwise. The pre-image fan is computed and drawn for each node at each successive level in the tree. The endpoints of the fan where the nodes are drawn are positioned on notional concentric circles around the attractor cycle corresponding to successive levels in the tree. The distance between each successive concentric circle, and thus between tree levels away from the root, decreases asymptotically for 90 levels, thereafter remains constant.

The tree will grow away from the attractor, backwards in time. Once the tree is complete with all its garden-of-Eden states reached, the next tree (anti-clockwise on the attractor) is computed. (note that with compression on (see section 26.1), equivalent trees will be computed and drawn simultaneously).

23.3.1 Transient tree colors

A cycle of four colors is used to draw transition arcs. The arcs in the same pre-image fan are drawn in the same color (except for the uniform states, see below). If the attractor period is 5 or less, successive fans are assigned a different color so that a given transient tree may have a mix of colors. For attractor periods of 6 or more, all fans in the same tree are assigned the same color, and the color is changed for the next non-equivalent tree. Note that with compression on (see section 26.1) equivalent trees will be colored identically. Conventions for bit pattern node colors are described in section 26.2.1.

23.3.2 Transient trees for the states all 0s or all 1s

If CA compression is set, a special algorithm is employed to speed up computation of the subtree of the "uniform" states consisting of all 0s or all 1s, when these are on the attractor (which must have a period of 1 or 2). If a given state is a pre-image of a uniform state, then that state's rotation equivalents[12] must also be pre-images, and may be computed simultaneously by an appropriate rotational transformation.

The first level of a uniform state's tree is organized into groups of equivalents (shown in different colors). The subtree of each representative state in each group at this first level is computed in turn. Each subtree will be completed before the next is started. Successive pre-image fans in each non-equivalent subtree are assigned a different color. Equivalent subtrees will be computed and drawn simultaneously, and will be colored identically.

Examples are shown in figure (f) in each of the figures 23.2, 23.3, 23.4.

23.3.3 Subtree only

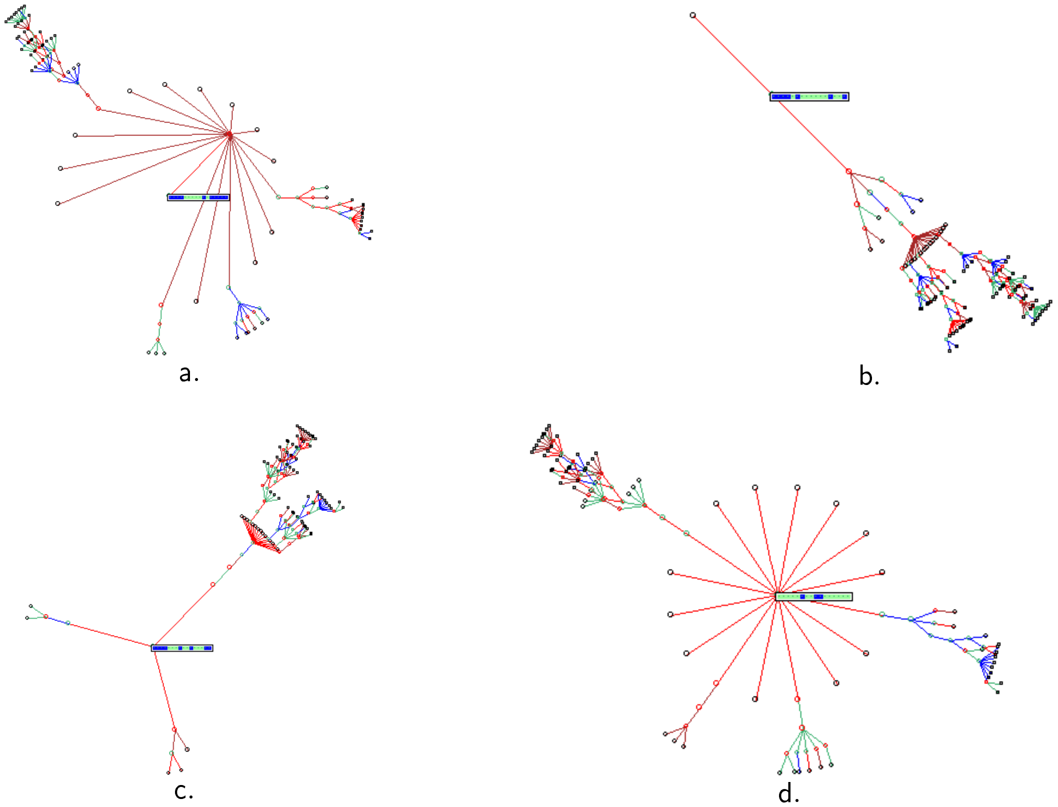


Figure 23.6: Examples of sub-trees from root states with in-degrees (at level 1) of 1, 2, 3 and 16 (n=16, k=3, rule 193). Note that the root state for a sub-tree is shown as a bit string by default.

A subtree only may be selected, running backward from a given state, or from a state a specified number of steps forward from a given state. The first pre-image fan is evenly spread around a notional circle with a diameter equal to the current maximum attractor diameter. Successive pre-image fans (assigned different colors) are computed and drawn for each node at each successive level in the subtree.

If the fan at level 1 has just 1 state, the fan at level 2 will be spread out as shown in figure 23.3a, similar to the the "pre-image fan of a single pre-image of a point attractor" (figure 23.3).

Chapter 24

Output parameters for attractor basins

The options in this chapter describe various output parameters for the display and analysis of attractor basins. In chapters 25 to 28 there are additional options for layout, display, data, and mutation. Yet another set of options are presented just before attractor basins are draw (see chapter 29).

Although there are many output parameter options and sub-options, all have defaults. The options may all be looked at in turn, but they are divided into five categories to allow jumping directly to the category where options need to be set, as follows,

revise from start-ret ...	miscellaneous options difficult to categorize.
layout-l ...	layout of attractor basins; their scale, position and spacing.
display-p ...	display of attractor basins and nodes.
data/pause-t ...	pausing between trees/basins, specifying data to be recorded, and setting the metagraph of the basin of attraction field.
mutation-m ...	various mutation settings for the *next* network. Its attractor basin may be generated immediately with the same settings after the current attractor basin is complete.

Enter **d** in section 24.1 to skip all these categories, accept the defaults, and go directly to the final prompts before attracor basins are drawn (see chapters 29 and 30). Its possible to see options one after another (with **return**), were one category leads to the next, or to skip between the categories, forwards (as described), and backwards (with **q**). Parameter changes become new defaults. This chapter describes the initial set of miscellaneous options that are difficult to categorize. Chapters 25 to 28 describe the other four categories.

Note that these options are skipped if space-time patterns only are selected by entering **s** in section 24.1. They are skipped automatically if the following option, for regular 2d or 3d CA wiring, was selected at the wiring prompt in section 10.6.1.

forwards only (uses less memory, slower)-y:

In this case the following reminder is displayed in a top right window,

2d CA (40 x 40), forward only, return to continue: *(values shown are examples)*

24.1 The first output parameter prompt

The first output parameter prompt for attractor basins is presented in a top right window as follows,

> **accept all basin defaults-d, space-time patterns only-s**
> **restore defaults: all-a, layout only-L**
> **revise from: layout-l display-p data/pause-t mutation-m**
> **revise from start-ret:**
>
> ..., **space-time patterns only-s**
> *(If a basin of attraction field was not selected in the first DDLab prompt, section 6.1)*

At the same time **output parameters** appears in a top center window.

option ...	*what it means*
accept all basin default ...	Enter **d** to accept all output parameter defaults, and skip further prompts. This can be done at any stage in the prompt sequence. New settings generally become the new defaults, but can be reset to what they were originaly. The next prompt for a basin of attraction field will be section 29.4, or for a subtree or single basin, section 29.1.
space-time patterns only-s ...	If a basin of attraction field was *not* selected at the first DDLab prompt (section 6.1), and space-time patterns (running forward) only are required , enter **s** to skip all attractor basin options and go directly to the output parameters for space-time patterns (see chapter 31). Note that a final chance to select space-time patterns is given in section 29.3.
restore defaults: ...	*options for resetting original default parameters*
all-a ...	Enter **a** to restore all defaults to their original settings.
layout only-L ...	Enter **L** to restore just the layout defaults (see chapter 25) to their original settings.
revise from: ...	*jumping directly to options described in chapters 25, 26, 27, 28.*
layout-l ...	Enter **l** to jump directly to a sequence of prompts to set the scale, position and spacing of attractor basins (see chapter 25).
display-p ...	Enter **p** to jump directly to a sequence of prompts to make changes to the way attractor basins are displayed, the compression of equivalent basins and trees, presentation nodes, orientation, and in-degree angle (see chapter 26).
data/pause-t ...	Enter **t** to jump directly to a sequence of prompts for pausing attractor basins, allowing the layout to be re-adjusted on-the-fly, to show data at various levels of detail, options to print and save this data, and to set the meta-graph and jump-table of a basin of attraction field (see chapter 27).
mutation-m ...	Enter **m** to jump directly to to a sequence of prompts to set the type of mutation to be applied to the *next* network (see chapter 28).

When the drawing of attractor basins is complete for a given network, another attractor basin may be immediately generated with the same settings, but with a mutation to some aspect of the networks wiring or rules.

revise from start-ret ... Enter **return** to start miscellaneous options which are hard to categorize, presented in sequence in a top right window. These are described in this chapter. At any point enter **q** (or **q** more than once) to backtrack to the first output parameter prompt. Remember that entering **d** will accept all remaining defaults, and skip further output parameter prompts.

24.2 State-space matrix

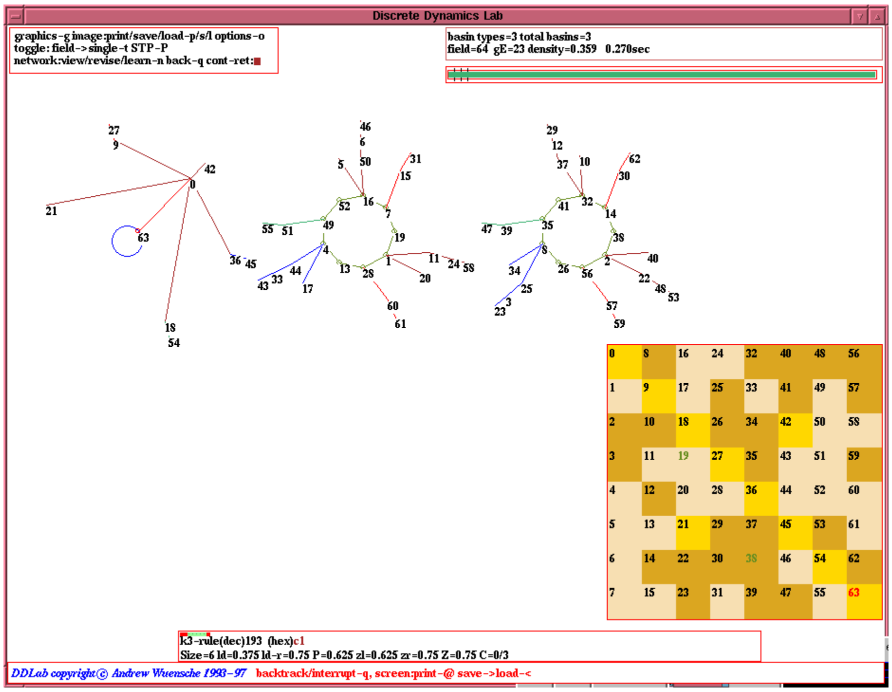


Figure 24.1: The state-space matrix (lower right) of the basin of attraction field of a cellular automaton, $k = 3$ (rule 193), $n = 6$. For n=6 the default grid size is large enough to show the the decimal equivalents of states, which are also shown in the state transition graphs (above). The 3 basins are represented by 3 colors in the state-space matrix.

The state-space matrix plots each state in state-space on a 2d grid in the lower right corner of the screen, plotting the left half against the right half of each state bit string. The x-axis represents the left $n/2$ bits, the y-axis represents the right $n/2$ bits. If n is odd, the extra bit is included on the left, and the grid is a flat rectangle as in figure 24.2(*bottom*), otherwise the grid is square. The scale of the matrix is set automatically according to n, though this can be changed.

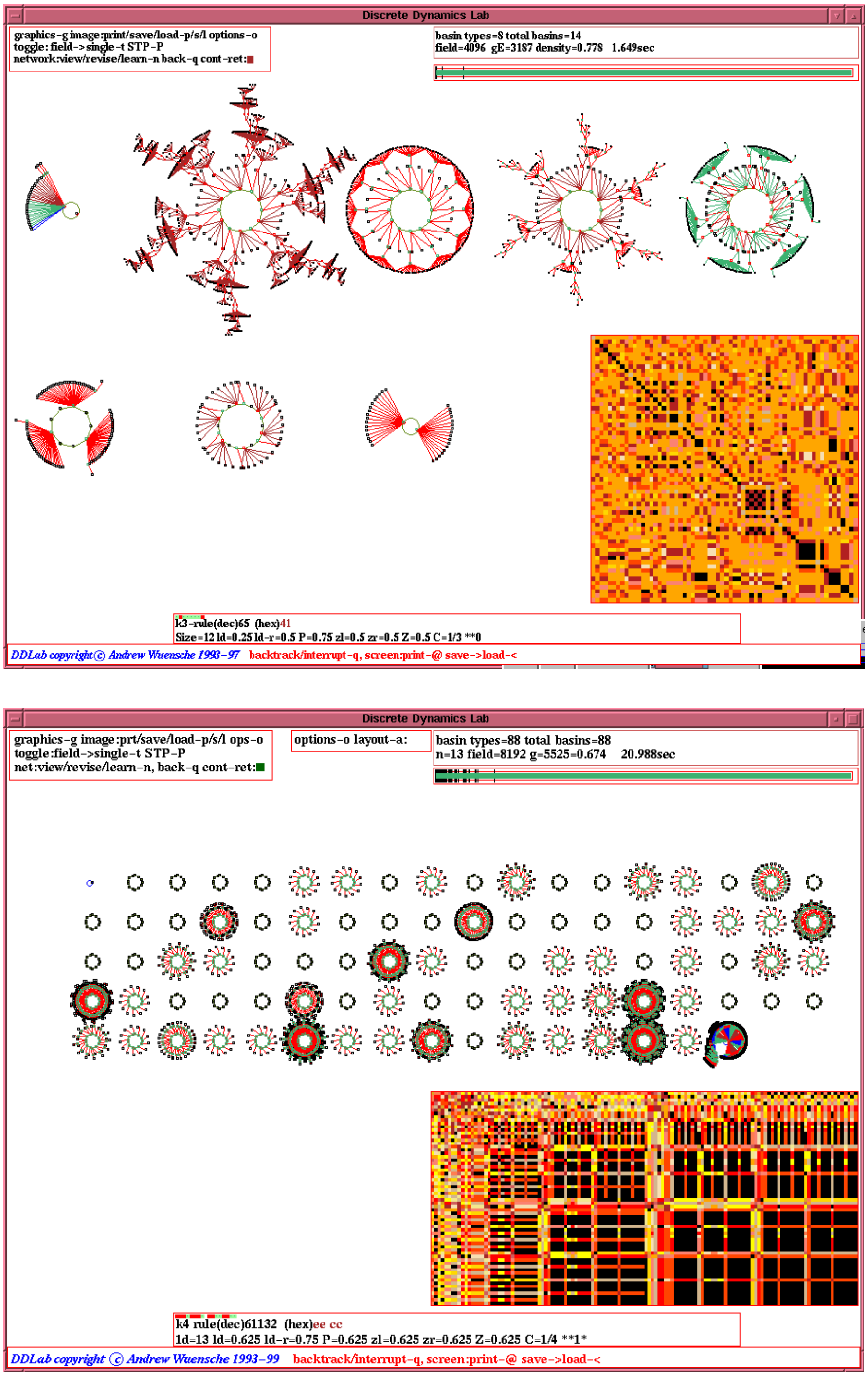
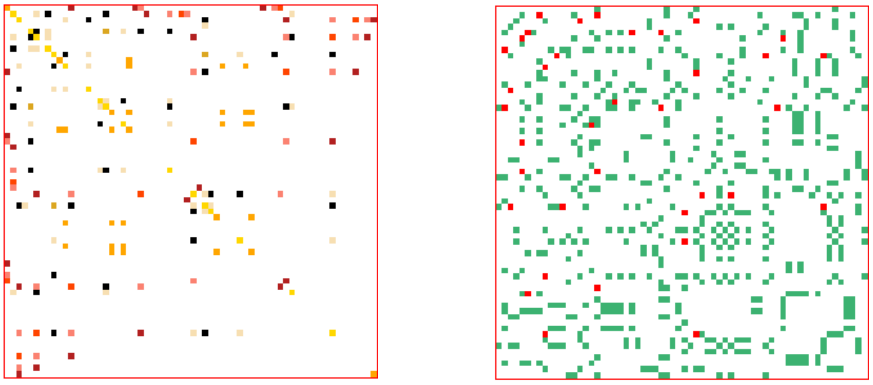


Figure 24.2: The state-space matrix (lower right) of the basin of attraction field of CA, with equivalent basins suppressed. *top* $k = 3$ (rule 65), $n = 12$, the 8 basins types are represented by 8 colors in the matrix. *bottom* $k = 4$ (rule ee cc), $n = 13$. Becase n is odd, and the extra bit is included on the left, the grid is a flat rectangle. The 88 basins are represented by cycling through 15 colors.



(a) attractor states only (b) a single basin

Figure 24.3: The state-space matrix of the same CA shown in figure 24.2(*top*), $n = 12$, $k = 3$, rule 65,
(a) just the attractor states of the of the 8 basin types represented by 8 colors.
(b) just the states in the 5th basin type (top right in figure 24.2(*top*)). The attractors and transient state are represented by 2 different colors.

For a single basin, attractor cycle states and transient states are shown in different colors. For a basin of attraction field, each basin is assigned a different color (cycling through 15 colors). The start color for the color cycle may be reset to produce different color schemes. If the grid size is big enough, the decimal equivalent of the states are also printed (see figure 24.3).

The following prompt is presented,

VARIOUS: state-space matrix: all-states-m, attractor only-a
show and change: matrix size-s, start color-c:

Note that the state-space matrix can also be toggled on-the-fly when showing space-time patterns, forward only (see section 32.11.7).

24.2.1 Toggle the matrix on-the-fly

Whether set or not in section 24.2, the state-space matrix can be toggled on/off while attractor basins, or just space-time patterns, are being drawn. The following reminder will appear in the bottom title bar,

matrix-m STP-s scroll-# exp-e contr-c

Enter **m** to toggle the state-space matrix on-the-fly. Options #, **e** and **c** are described in section 24.7.1.

24.2.2 Display all states

Enter **m** in section at 24.2 to display the matrix, including all states in the subtree, single basin or field. For a single basin, attractor cycle states and transient states are shown in different colors. For a basin of attraction field, each basin is assigned a different color (cycling through 15 colors).

24.2.3 Display attractor states only

Enter **a** in section 24.2 to display the matrix showing just attractor states in the single basin or field. For a basin of attraction field, each basin is assigned a different color (cycling through 15 colors).

24.2.4 Change matrix size

Enter **s** in section 24.2 to change the matrix size from the default. The following prompt is presented,

change matrix size: % of 1/2 screen(default 63%):

Enter the new size as a percentage of half the screen width. If the matrix size is big enough, the decimal equivalent of states will be shown on the matrix.

24.2.5 Change start color

Enter **c** in section 24.2 to change the start color in the color cycle of 15 colors, and produce different color schemes. The following prompt is presented,

change start color (now 10): enter 1-15:

24.3 In-degree frequency histogram

DDLab may be set to keep a record of the frequency of different in-degrees (the number of pre-images or predecessors) that occur in a subtree, basin or field. If this option is selected, the in-degree information is displayed as a histogram when the attractor basin is complete, or during a temporary pause (see #25.1). The following prompt is presented,

show pre-image frequency histogram-y:

If **y** is entered, the histogram setting remains active for all further attractor basins. It may be deactivated by not entering **y** at this prompt, by restoring all "output parameter" defaults (see section 24.1), or by backtracking through the main prompt sequence beyond section 8.

24.3.1 In-degree frequency cut-off

If **y** is entered above, the following prompt allows setting an upper threshold (or cut-off) to the in-degrees recorded, in order to conserve memory. Any cut-off in the range of 50 to 5000 may be set, the default is 200, or the current setting. The occurrence of in-degrees equal to or above the cut-off value will be added to the cut-off frequency "bin" in the histogram.

select cut-off in-degree bin (50-5000, default 200):

A further prompt, **show/count states with majority of: 1s-1, 0s-0:** is descussed in section 24.3.9.

24.3.2 In-degree frequency window

The in-degree histogram will be displayed in a window in the lower section of the screen. The x-axis represents the range of in-degrees, from in-degree zero (garden-of-Eden states) upward. In-degrees equal to the cut-off value and above are combined and represented by the last frequency column. The y-axis represents the number of states having a given in-degree. The cut-off value column (if included) is represents excess in-degrees in black. The garden-of-Eden column is colored red, the other columns yellow. Initially the scales of the x and y axis are automatically set according to the range of in-degrees encountered, but can be separately re-scaled or shown in log form for a clearer view of the spread of frequencies or to amplify smaller in-degrees.

24.3.3 In-degree data and prompts

An example of the data displayed in the window, and initial prompts (as in figure 24.4 for the basin of attraction field of a k=3 CA, rule 193, n=18) is shown below,

total att/nodes/edges=487/262144/262144 *(for a subtree* **subtree: nodes/edges....)**
max in-degree=158, gE=161096 *(values shown are examples)*
log x-x y-y both-b, save/load data-s/l, re-scale-ret, quit-q:

If **return** is entered above further options and rescaling promps are presented,

(options-o prt-p quit-q) *(for DOS* ..**prtbox-p**..)
re-scale x/y axis (def-d): x-max (30-158): y-max:
*(***x-max** *only if the max in-degree found is greater than 30)*

24.3.4 Decode of data shown in the in-degree window

The data shown in in-degree frequency window (section 24.3.2) signifies the following,

att ...	the number of attractor states (does not apply to subtrees).
nodes ...	the number of nodes whose pre-images have been computed.
edges ...	the total number of edges computed so far.
max in-degree ...	the largest in-degree found. The maximum equals the cut-off limit.
gE ...	the frequency of garden-of-Eden states (in-degree 0), usually the greatest frequency.

Note that for a complete basin (or field), nodes=edges, for a complete subtree, nodes=edges+1, and for an incomplete attractor basin, (i.e. if interrupted), nodes<edges.

24.3.5 In-degree log plot

Enter **x**, **y** or **b** in section 24.3.3 to replot the histogram with the x-axis, y-axis or both axies showns according to log scale base 2.

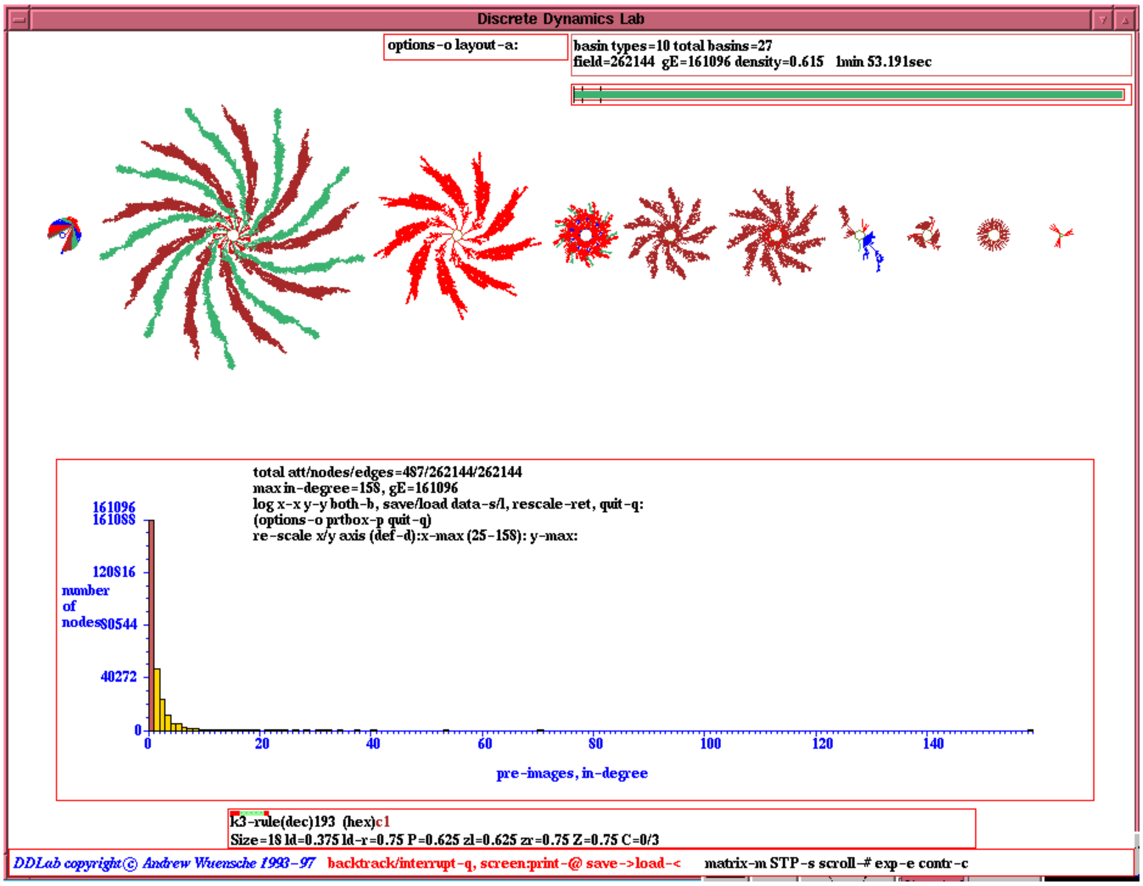


Figure 24.4: In-degree histogram (below) of a a basin of attraction field of a CA, $k = 3$ (rule 193), $n = 50$. Attractor basins are shown above.

24.3.6 Save/Load in-degree data

Enter **s** to save the in-degree histogram data to a `.his` file, or **l** to load a `.his` file and see it displayed as a histogram (see Filing, chapter 35).

Note that when an in-degree histogram data file is loaded, the **total att/nodes/edges** and **max in-degree** data is not displayed, but the histogram can be shown in log form or rescaled as described. An example of the initial in-degree data and prompts for loaded data (as compared with section 24.3.3) is as follows,

> **data loaded**
> **gE=3952** *(values shown are examples)*
> **log x-x y-y both-b, save/load data-s/l, re-scale-ret, quit-q:**

24.3.7 Rescaling the x/y-axis

To rescale the x and y axies enter **return** in 24.3.3, then enter the new maximum values at the next prompt, or **d** to restore the axies scale to their original settings.

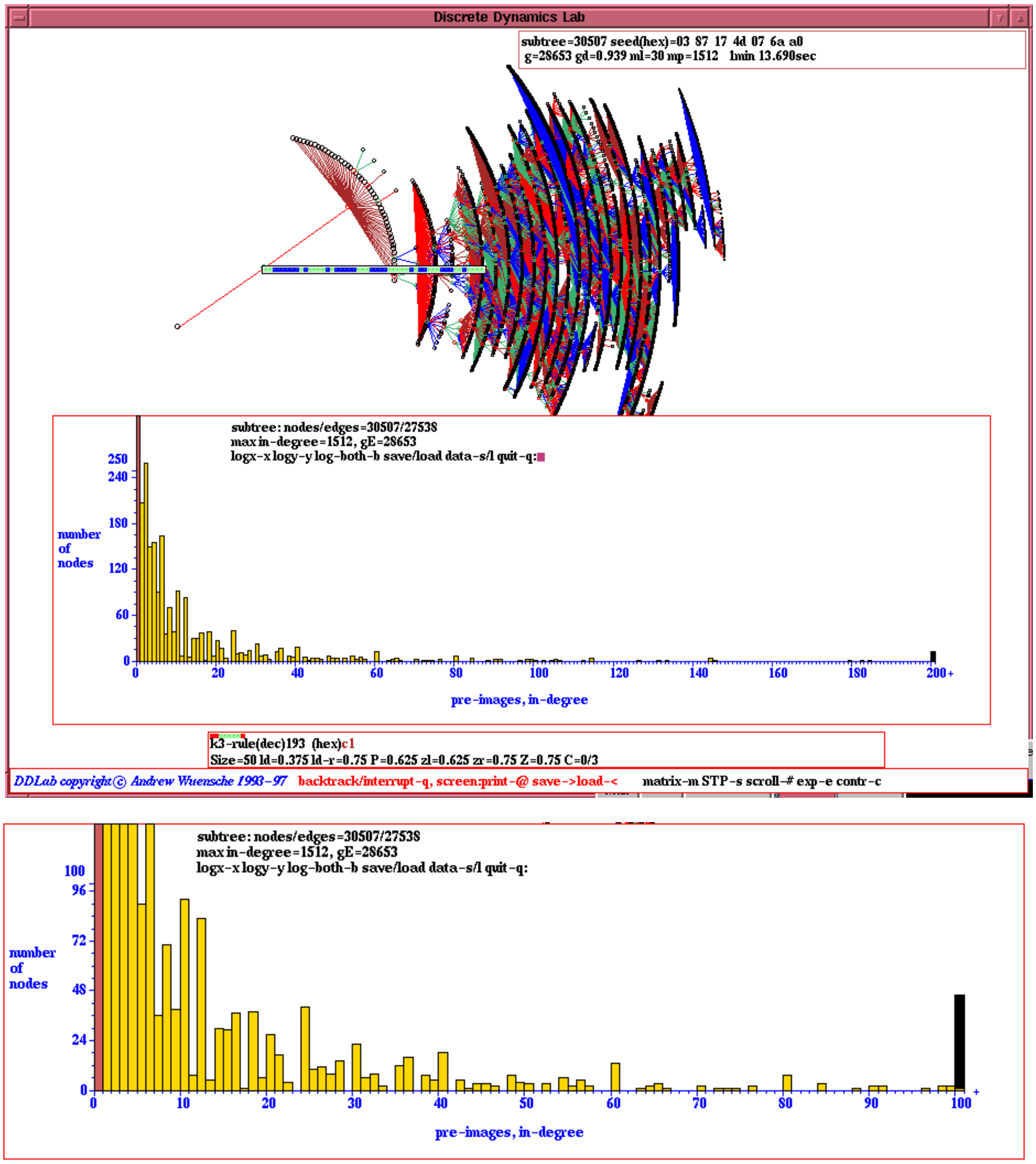


Figure 24.5: *above* A subtree of a CA, $k = 3$ (rule 193) with the root state indicated, and in-degree histogram. *below* The same in-degree histogram, but with the x and y axis rescaled for a closer view. Excess in-degrees (indicated by "+") are included with the last black column on the right. Zero in-degrees (garden-of-Eden states) are in the red column on the left.

The x-axis is initially scaled to show the spread of in-degrees from zero to the maximum found (or to the cut-off value). If the cut-off value is shown, + is added at the end of the x-axis indicating that in-degrees equal to or greater than the cut-off value occurred.

If the maximum in-degree found is greater than 30, the x-axis can be rescaled by entering a new maximum in-degree. Excess in-degrees (off the scale) will be shown as a black column on top of the maximum in-degree column, and + will be added to the x-axis.

The y-axis is initially scaled to contain the highest in-degree frequency column in the histogram (usually column 0, garden-of-Eden states). It may be rescaled to any value, for instance to show smaller in-degree frequencies at a larger scale.

24.3.8 Further in-degree prompts

At the prompts in 24.3.8, enter **o** for further options described in section 30.4, enter **p** to print just the in-degree window (DOS only), or **return** to return to the initial in-degree prompts (section 24.3.3).

Enter **q** either here or in section 24.3.3 to quit the in-degree window and uncover any attractor basins which may have been hidden by it. However, if in-degree histogram data was loaded (see section 24.3.6) the original in-degree window and its data will be restored.

24.3.9 States with majority of 1s or 0s

A further prompt in section 24.3.1 allows states with a either a majority of 1s or 0s to be highlighted in attractor basins, and for subtrees the "exceptions" in the majority problem are given.

The following prompt is presented,

show/count states with majority of: 1s-1, 0s-0:

Enter **1** or **0** to highlight these states. The highlighted states are displayed as small yellow squares with a black border (figure 24.4). This option is included to test the performance of a favourite exercise in "emergent computation", where CA rules are evolved by a genetic algorithm to discriminate between states with a majority of 0s and 1s. This usualy entails evolving a rule with two point attractors, the all 0 state attracting the majority-0 states, and the all-1 state attracting the majority-1 states. If this option is selected, for a subtree only, the number and percentage of "exceptions" to either of these conditions is displayed in the in-degree frequency window.

24.4 "Screen save" demo

If a single basin or subtree was selected at the first prompt in section 6.1, an option is presented for a continuously changing display where basins or subtrees are generated randomly in overlapping bubbles on the screen (see also section 4.11 in the Quick Start Examples. The following prompt is presented,

screensave demo -s:

Enter **s** to select the demo. An example is shown in figure 4.12. The backwards space time patterns and state-space matrix may also be set, or toggled on-the-fly (see sections 24.2.1 and 24.7).

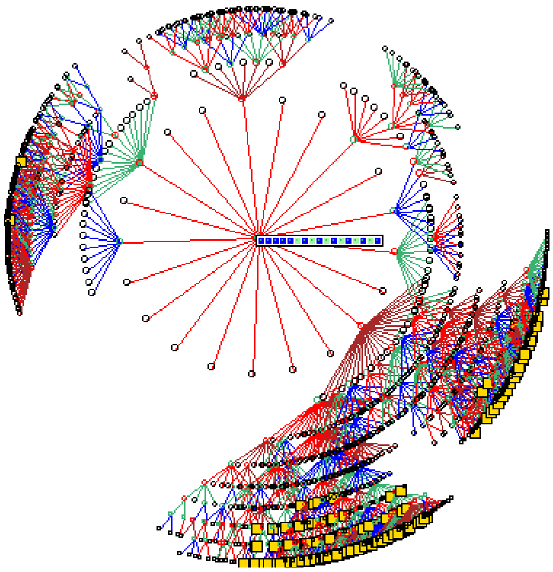


Figure 24.6: A subtree showing exceptions in the majority problem. In this example a subtree is generated from an initial state, n=17, with 11 1's, a density of about 0.65. The rule is one that was evolved to discriminate between states with a majority of 0s and 1s, the k=7 rule ff f0 8c f0 ff e0 00 f0 ff f0 00 f0 ef e0 00 e0. The states in the subtree should also also have a majority of 1s, but there are exceptions, about 11.6%, indicated in the in-degree frequency window, and by yellow squares with a black border in the subtree.

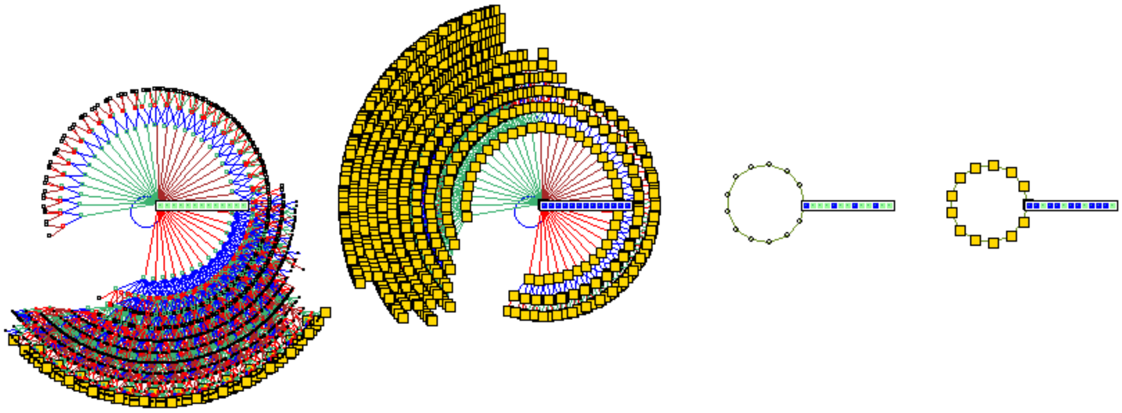


Figure 24.7: A basin of attraction field showing states with a majority of 1s. n=13, k=7, the rule is the same as in figure 24.6. There are 4 basins, but nearly all states go to the all 0's and all 1's point attractors. States with a majority of 1's are indicated in the basins by yellow squares with a black border.

24.5 Graphs: G-density, Z and λ

The density of garden-of-Eden states[1] in a subtree, basin or field (G-density) is a measure of the convergence in network dynamics, which in turn relates to the quality of typical space-time patterns, ordered-complex-chaotic in CA[19]. The λ parameter (or the equivalent P parameter) is a measure of the fraction of 1s/0s in a rule-table, whereas Z measures the probability that the next cell in a candidate partial pre-image of a CA is determined. Both of these related parameters predict convergence and thus the quality of the dynamics to varying degrees, though Z is more focused. The λ_{ratio} is the fraction of 1s or 0s (whichever is less) in relation to half the rule-table, normalizing the λ parameter to a value between 0 and 1 for direct comparison with Z.

[1] Garden-of-Eden states are those without predecessors

In order to test these relationships, the following sections describe how graphs of G-density against Z and/or λ_{ratio} may be plotted for a predefined set of CA, and graphs of G-density against a range of system size n. Graphs of Z against λ_{ratio} may also be plotted to relate the two parameters.

In addition, the proportions of rules in rulespace with various levels of "canalizing inputs" and the P parameter may be computed. These measures are of interest in predicting convergence and stability in random Boolean networks.

The following prompt is presented,

G-density graph: Zpara-1, and Lambda ratio-2, size range-3
C-P data, Zpara-Lda ratio, Zleft-Zright graphs -L:

24.5.1 G-density against Z and/or λ_{ratio}

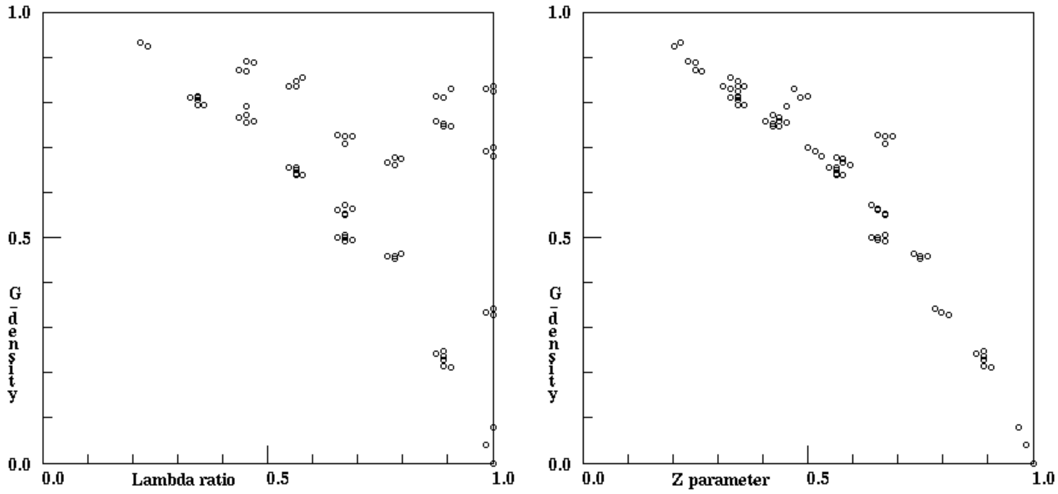


Figure 24.8: G-density plotted against λ_{ratio} (left) and the Z parameter (right). n=12, for the k=7 totalistic rules. The complete basin of attraction fields was generated for each rule and the garden-of-Eden states counted.

Enter **1** in section 24.5 for a graph of G-density against Z, or **2** for an additional graph of G-density against λ_{ratio} drawn simultaneously.

Enter **d** to skip further options. The Z graph will be drawn in the lower right of the screen, the λ_{ratio} graph beside it on the left. A subtree, basin or field for each of the set of rules specified will be drawn (at a small scale by default) in a window in the upper part of the screen (see figure 24.9. When each attractor basin is complete the G-density (y-axis) is plotted against the rule's Z and/or λ_{ratio} on the x-axis.

24.5.2 G-density for totalystic rules, or $k \leq 3$

If the rules specified are totalistic rules, or if $k \leq 3$, the rules included in the plot consist of a countdown from the decimal rule number or totalistic code originaly selected in section 16. To include all rules the maximum decimal rule number or totalistic code should have been selected.

However, the range of Z may be restricted, with a lower and upper limit. Note that low values of Z (especially Z=0) should be avoided except for small network sizes, because in-degree (and G-density) will be very high. Very high in-degrees can exhaust computer memory or take a long time to compute.

The following prompts are presented, firstly to set the lower limit, then the upper.

min Z (def 0.2): **set max Z (def 1):**

24.5.3 G-density for non-totalystic $k > 3$ rules

If the rules specified are not totalistic rules and if $k > 3$, the following prompt is presented,

G-density graph: number of rules (def 1250):

Enter the number of rules to be plotted or accept the default. The rules will be selected at random, but with a bias to include a representative distribution over values of Z.

24.5.4 G-density plotted against network size

The rate of increase of G-density with system size n indicates order-disorder in a CA, with ordered dynamics showing the highest G-density and rate of increase[19].

Enter **3** in section 24.5 to plot G-density against a range of n. If required, this can be done for a number of selected rules on the same plot. The following prompt is presented,

G-density - size range graph
start size (def=5): **end size (max=31):**

Enter the start and end values of n. Note that if the end size is set high, computation time may be inordinately long.

Enter **d**, which normaly skips all further options, but in this case skips to mutation (see chapter 28). The graph will be drawn in the lower right of the screen. The basin field will be draw (at a small scale by default) in a window in the upper part of the screen. When each attractor basin is complete the G-density (y-axis) is plotted against the start value of n, and the plot will continue automaticaly for the next value of n. Once complete, the following prompt appear in the top right data winfow,

quit-q rule-r (mut-def):

Enter **r** to select a new rule that will be superimposed on the plot, **return** for the next mutant, according to the type of mutation set in section 28.1, or enter **q** to quit the G-density - network size graph, and backtrack.

24.5.5 Z - λ_{ratio} graph

Enter **L** in section 24.5 to plot a graph of Z (x-axis) against λ_{ratio}, or Z_{left} against Z_{right}, for a range of rules. The graph will be plotted in the lower right part of the screen. In addition, the proportions of rules in the rule sample with different levels of "canalizing inputs" and the P parameter may be displayed once the graph is complete.

The measures are computed directly from the rule-table. The following prompt is presented,

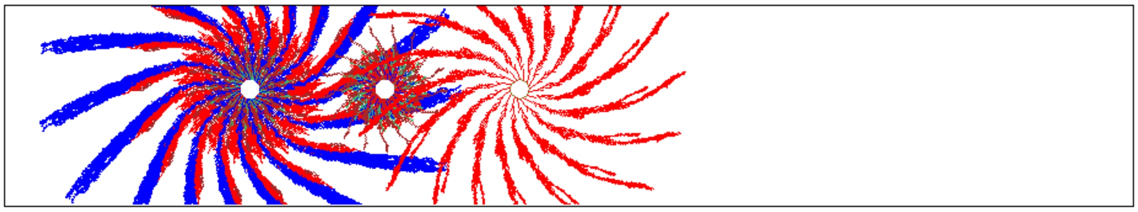


Figure 24.9: As the G-density - network size graph is plotted (and also graphs in section 24.5), the attractor basins are shown in an upper window at a default scale and layout (which can be changed in chapter 25). This example shows the CA basin of attraction field for rule 99 4a 6a 65, for n=17, to plot the final point of the lower graph in figure 24.10.

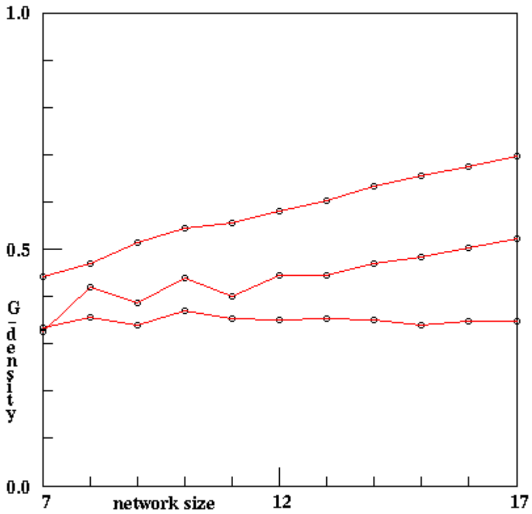


Figure 24.10: Plotting G-density against a range of n for ordered, complex and chaotic CA rules, where the ordered rule has the highest G-density. Rules (from the top) 01 dc 36 10, 6c 1e 53 a8 and 99 4a 6a 65.

draw Zleft-Zright -1, or Zpara - Lda ratio (def):

Z_{left} and Z_{right} are components of the Z parameter. Whichever graph is selected, the prompts that follow are similar.

Enter **return** for a Z - λ_{ratio} plot, or **1** for a Z_{left} - Z_{right} plot.

24.5.6 Z - λ_{ratio} or Z_{left} - Z_{right} for totalystic rules, or $k \leq 3$

If the rules specified are totalistic rules, or if $k \leq 3$, the rules included in the plot consist of a countdown from the decimal rule number or totalistic code originaly selected in section 16. To include all rules the maximum decimal rule number or totalistic code should have been selected.

For totalistic rules the following prompt is presented, allowing "equivalent" rules to those already plotted to be suppressed,

show only equivs-e all-a (def clusters): *(totalistic rules)*

Enter **a** to show all totalistic codes (see section 12.2) less than or equal to the code originaly selected in section 16, or **e** to show a representative code from each set of equivalent codes. The default is to show a representative rule from each rule cluster. There are at most 2 equivalent totalistic rules, the start rule and its negative (the reflection of a totalistic rule = identity), and at most 4 rules in a totalistic rule cluster which includes the compliments of both equivalents[12] . Rules in a rule cluster have the same measures of Z, λ_{ratio} (and also G-density) so the resulting plots will be identical. A further prompt allows a pause to note the equivalents or members of the cluster,

pause to show codes-p: *(totalistic rules)*

If **p** is entered above, after each point is plotted, data about equivlent rules and clusters will be displayed in a top right window, for example for $k = 7$,

7-code:233=104 22=151 equiv=2 clust=4 tot=22 cont-ret:
(values shown are examples)

This signifies that codes 233 and 151 are equivalent as are their compliments, codes 22 and 151. These 4 codes make up a cluster. In this example, the total clusters plotted so far is 22.

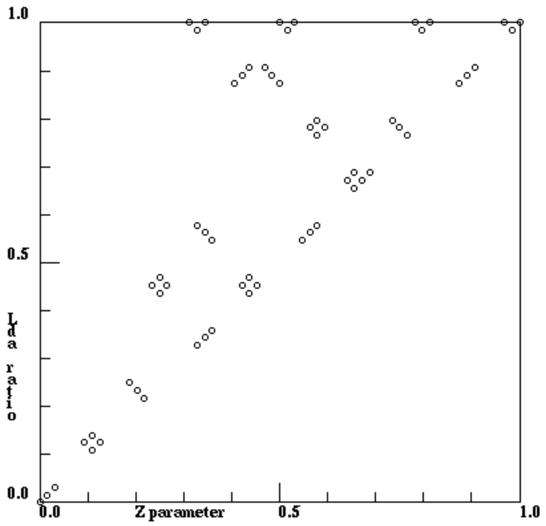


Figure 24.11: λ_{ratio} plotted against Z for the 256 $k = 7$ totalistic rules. Because of rule-table symmetries this number reduces to 136 non-equivalent rules belonging to 72 clusters having equal Z, λ_{ratio} and G-density

24.5.7 Z - λ_{ratio} or Z_{left} - Z_{right} for non-totalystic, $k > 3$ rules

If the rules specified are for $k > 3$ and not totalistic, the following prompt allows the rule sample size to be specified,

lambda-Z graph, number of rules (def 2200):
or
Zleft-Zright graph: number of rules (def 2200):

By default the rules will be selected at random, but with a bias to include a representative distribution over values of Z. The following further option allows an unbiased random sample.

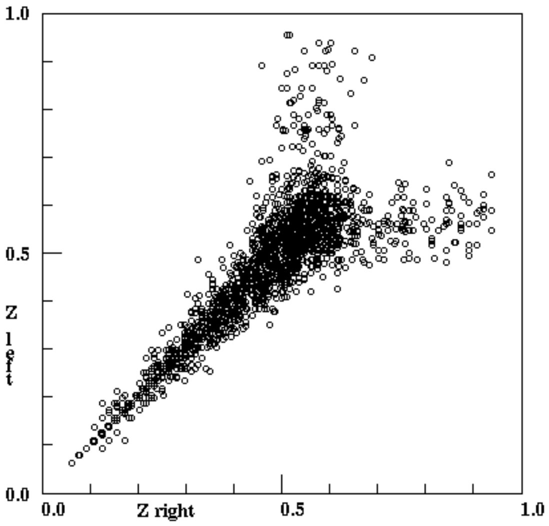


Figure 24.12: Z_{left} plotted against Z_{right} for 2200 $k = 7$ rules, selected at random but with a bias to cover a representative spread of Z.

random rule sample will have lambda bias, dont bias-n:

An unbiased sample is useful in estimating the proportion of rule-space with different measure of the P-parameter and canalizing inputs (see section 24.6 below).

24.6 Proportions of canalizing inputs and λ_{ratio}-P

Once the Z - λ_{ratio} graph is complete, data on the proportions of rules in the rule sample with different levels of "canalizing inputs", and the equivalent measures λ_{ratio} and P, may be displayed. The following prompt is presented in a small window to the left of the graph,

tot plotted=20000 *(for example)*
C-P data-d, +print-p: *(***+print-p** *not for DOS)*

Enter **d** to see the data table displayed across the top of the screen, (see figure 24.13 for a $k = 4$ example). Enter **p** to also print the data to the xterm window (not for DOS).

The data is decoded as follows,

Canalizing freq (C0, C1,...) ... the number and percentage of the possible canalizing levels, for k=4 there are 5 (C0-C4). (numbers are only shown if less than 9999)

Crules ... the number and percentage of rules with at least 1 canalizing input.

Cinputs ... the number and percentage of canalizing inputs in the rule sample.

ldr (first line) ... the possible values of λ_{ratio}, as a fraction, for k=4 there are 9 (0/8 to 8/8).

ldr (second line) ... as above shown as a fraction. λ_{ratio}= the fraction 0s or 1s (whichever is less) relative to half rule-table size, λ_{ratio} varies from 0 (order) to 1 (chaos).

P . . . as above showing the equivalent values of the P parameter. $P=$ the fraction of 0s or 1s in the rule-table (whichever is more), P varies from 1 (order) to .5 (chaos), and is related to λ_{ratio} as follows, $P = 1 - (\lambda_{ratio}/2)$

freq . . . the number and percentage of rules in each P or λ_{ratio} category above.

C=0, C=1,... . . . the number and percentage of each canalizing level in each P or λ_{ratio} category above (numbers are only shown if less than 999).

Canalyzing–lambda ratio and P frequency count, total rules=20000
Canalyzing freq: C0=94.660% C1=927=4.635% C2=114=0.570% C3=20=0.100% C4=7=0.035% total: Crules=1068=5.340%, Cinputs=1243=1.554%

ldr	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8
ldr	0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
P	1	0.9375	0.875	0.8125	0.75	0.6875	0.625	0.5625	0.5
freq	0	7=0.035%	69=0.345%	326=1.630%	5.545%	13.030%	24.625%	35.050%	19.740%
C=0	0	0	4=0.020%	109=0.545%	742=3.710%	11.824%	24.014%	34.823%	19.719%
C=1	0	0	18=0.090%	142=0.710%	355=1.775%	241=1.205%	122=0.610%	45=0.225%	4=0.020%
C=2	0	0	27=0.135%	75=0.375%	12=0.060%	0	0	0	0
C=3	0	0	20=0.100%	0	0	0	0	0	0
C=4	0	7=0.035%	0	0	0	0	0	0	0

Figure 24.13: Canalizing, λ_{ratio} and P data. An example For a $k=4$ random sample of 20,000 rules. The decode of the table is shown in section 24.6.

24.7 "Backwards" space-time patterns

At the same time that attractor basins are being drawn, the "backwards" space-time pattern may be displayed on the left of the screen.

For basins of attraction, this will show an initial run forward, representing a transient and the attractor cycle itself. For a just a subtree, an option allows running forwards for a given number of steps before running backwards (see section 29.2), which is usually necessary (other than for very chaotic networks such as CA with "chain" rules, see section 16.7) to avoid a garden-of-Eden state which has no pre-images thus no subtree. The space-time pattern of the initial transient in both cases is shown in blue and grey representing 1s and 0s.

Thereafter the space-time pattern shows the information used to draw each transient tree in turn, starting from the "root" of the tree (a state on the attractor if for a basin of attraction), and back through successive levels in the tree.

First the pre-images of the root are computed (excluding the pre-image on the attractor cycle if for a basin of attraction), comprising level 1 in the tree. Then the pre-images of each state in level 1 are computed, comprising level 2 in the tree. This process continues through an arbitrary number of levels until all garden-of-Eden states (those without pre-images) are reached. For a basin of attraction the next tree is then computed.

The "backwards" space-time patterns show each state (black and grey representing 1s ad 0s) and its set of pre-images if any (red and grey representing 1s ad 0s). The scale depends on the setting in section 7.3 in the main prompt sequence, but this may be changed on-the-fly (see below).

For a basin of attraction field the initial default is not to show the backward space-time patterns, and the following prompt is presented, enter **y** to show,

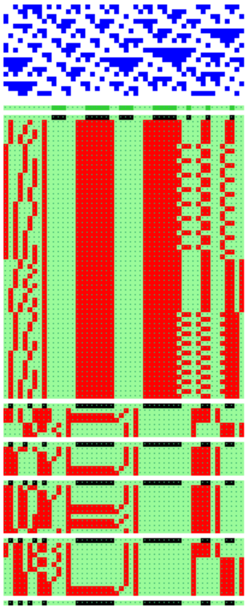


Figure 24.14: "Backwards" space-time patterns for a subtree. $k=5$ rule (hex) e0 89 78 01, n=50. The blue and white sites (1s and 0s) represent an initial forward transient of 20 time-steps. Then each state is shown in black and white with its pre-images (red and white) stacked below it. The figure shows just the start of the backward run, which may continue for an indeterminate length of time.

SHOW space-time patterns-y:

For a single basin or subtree, the initial default is to show the backward space-time patterns, and the following prompt is presented, enter **n** not to show,

NO space-time patterns-y:

24.7.1 Changing space-time pattern settings on-the-fly

Backwards space-time patterns can be toggled on/off while attractor basins are being drawn. They can be scrolled (see section 24.7.2) and the scale can also be changed. The following reminder will appear in the bottom title bar:

matrix-m STP-s scroll-# exp-e contr-c

Enter **s** to show/hide space-time patterns, **#** to toggle scrolling, **e** to expand and **c**the scale. Expanding/contracting the scale also affect the scale of the attractor basin nodes if displayed. Enter **m** to toggle the state-space matrix on-the-fly. Option **m** is described in section 24.2.1.

24.7.2 Scroll space-time patterns

By default, space-time patterns (both forward and backward) are presented in successive vertical sweeps, When the pattern reaches the bottom of the screen it continues from the top. Space-time patterns may alternatively be set to scroll, though this reduces the speed of the program. If space-time patterns were set in section 24.7, the following prompt is presented,

SCROLL space-time pattern-s:

Enter **s** to scroll. Scrolling can also be toggled on/off on-the-fly as described in 24.7.1

Chapter 25

Layout of attractor basins

Basins of attraction are drawn either singly, or in groups (for the basin of attraction field, for a range of network size n, or for mutant single basins). Groups of basins are drawn in successive rows starting from a top left position. Before drawing starts, the following layout parameters may be altered (or defaults accepted): their scale, position, angle, horizontal and vertical spacing, spacing increase (for a range of network size n), and the minimum right border before the next row starts.

A basin of attraction field may be displayed as successive single basins as well as in a group. Initial default layout values depend on the type of attractor basin selected. Once revised, the new values generally become defaults, but may be reset to the original values.

If **l** is selected at the first "output parameter" prompt in section 24.1 (or if the output parameter prompts are viewed in sequence) a layout preview for attractor basins will appear, together with the first layout prompt described below in section 25.2. Some layout parameters may be also be varied on-the-fly, after each subtree or basin is drawn, described in section 25.3.

The basin of attraction field can also be redraw within the attractor meta-graph window, according to the meta-graph layout (see sections 27.8) and 20.7, providing many additional layout possibilities to those described in this chapter.

25.1 The layout preview

The layout preview indicates the layout, spacing and scale of attractor basins as they would appear on the screen according to the current default settings. This is shown in a window representing the screen at 1/4 size, in the top left quadrant of the screen. Polar coordinates are used to scale and position basins, and these are indicated. x and y axes divide the screen with coordinates 0,0 at the center. At the left of the screen $x=$ -100, at the right $x=$ +100. At the top of the screen $y=$ +100, at the bottom $y=$ -100.

The layout preview represents the basins that might be drawn by concentric circles, where the inner circle is the maximum attractor diameter. Successive circles indicate successive levels. The first basin representation shows the first 90 levels where the distance between successive levels decreases asymptotically (thereafter the distance between levels remains constant). In the first basin every 20th circle is highlighted, as well the attractor circle itself and the first 3 levels. Other basin representations show the attractor circle and the first 3 levels only.

If a single basin or subtree was selected (without a range of network size n) only one basin preview is shown. Otherwise a representative number of basins are shown to anticipate the layout.

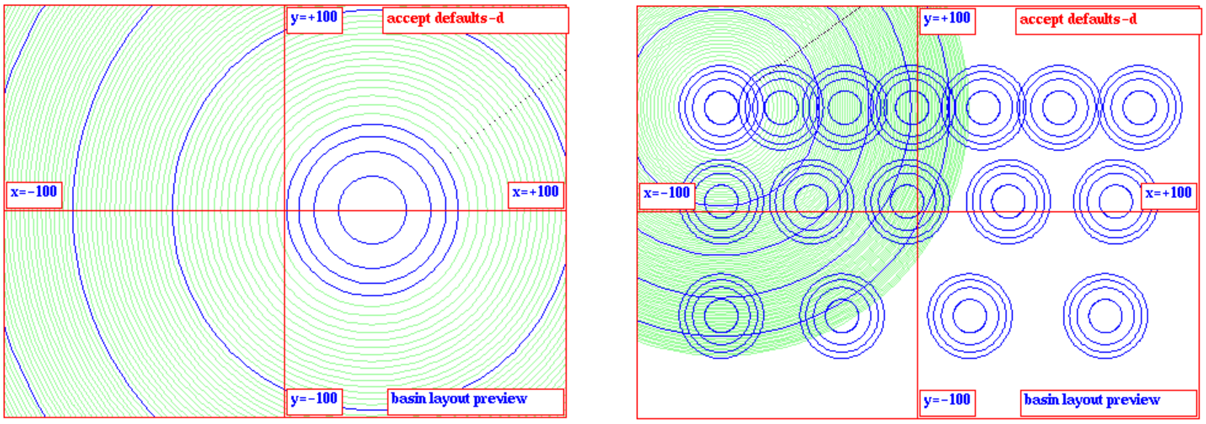


Figure 25.1: The default layout preview for a single basin or subtree (left), and for a range of single basins (right).

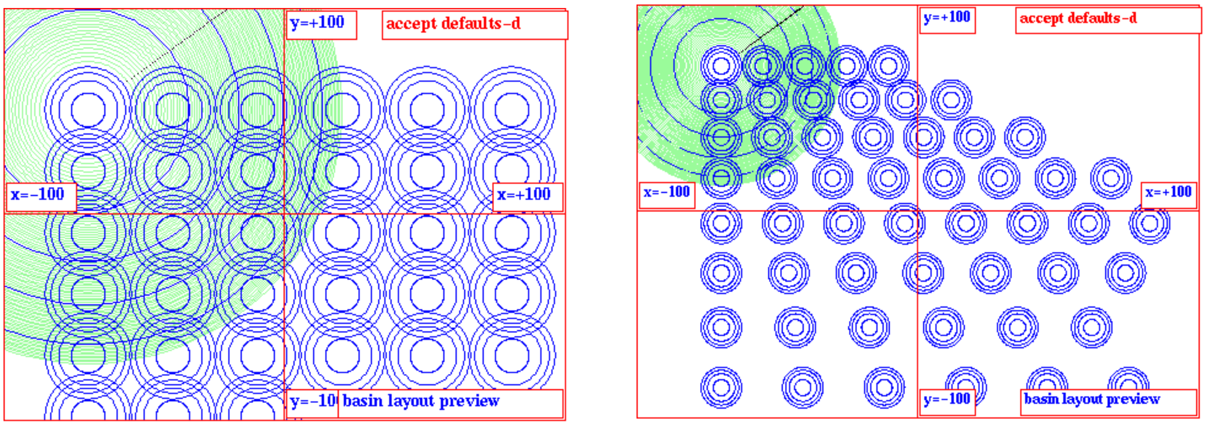


Figure 25.2: The default layout preview for a basin of attraction field (left), and for a range of basin of attraction fields (right).

25.2 The first layout prompt

Enter **l** at the first "output parameter" prompt in section 24.1 to skip directly to the preview for the layout of attractor basins (section 25.1). Alternatively these options can be viewed in sequence.

The layout preview for attractor basins will appear (described above, section 25.1), together with the following first layout prompt,

> **LAYOUT: reset defaults: all-r, learning-L mutants-M** *(***M** *for single basins only)*
> **select att rad, % of 1/2 screen, x-axis (min 0.25 def 6):**
> *(min depends on screen resolution, the default depends on previous selections)*

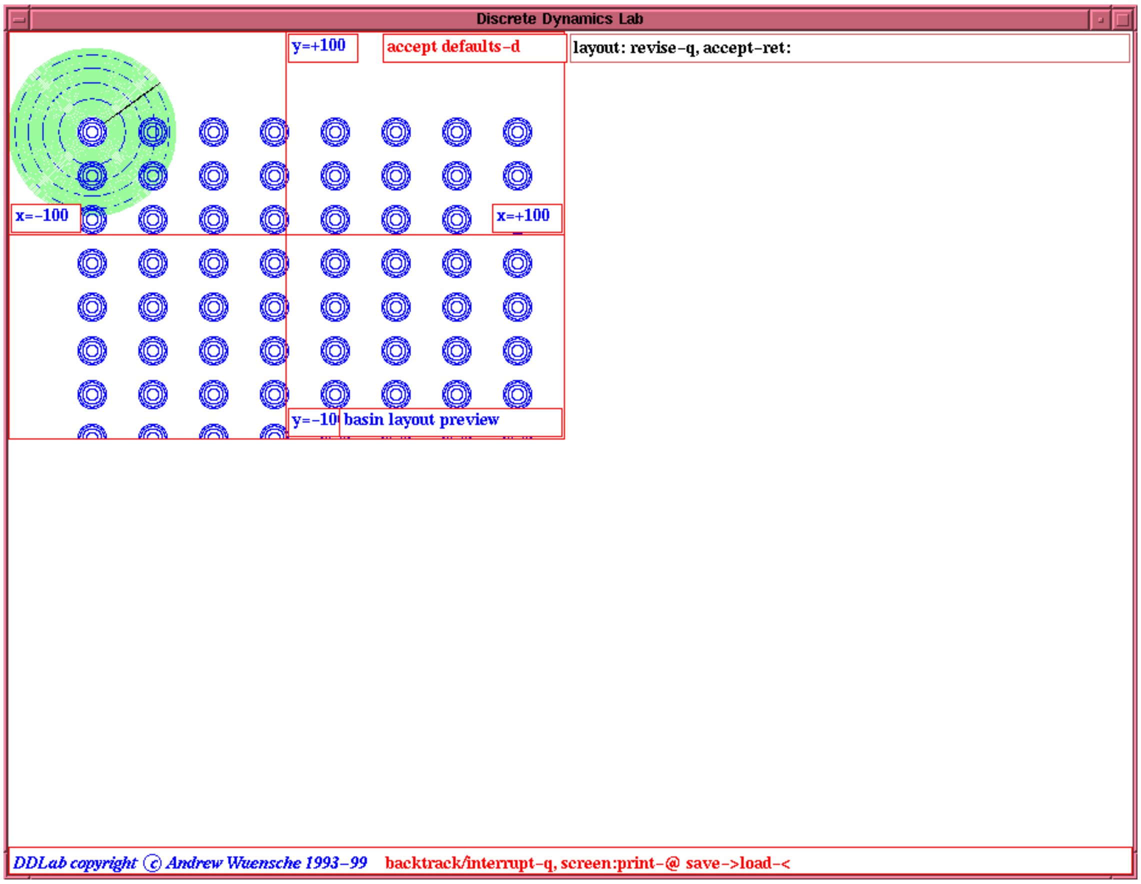


Figure 25.3: The layout preview as it appears on the DDLab screen. This examle is for a basin of attraction field, rad=2, x spacing=22, y spasing=22, right border=22.

25.2.1 Reset all layout defaults

Enter **r** in section 25.2 to restore all layout prompts to their original settings. These default settings depend on the type of attractor basin selected. The revised layout preview will be displayed.

25.2.2 Reset layout for learning

Enter **L** in section 25.2 to choose a default layout appropriate for "learning" (see chapter 34). A further prompt to set the node display is presented (see also 26.2),

> **nodes: none-n 2d-B 1d-b hex-h dec-c spot-s (def-s):**
> *(***dec-c** *if* $n \leq 32$*,* **2d-b** *for a 2d or 3d network)*

The attractor basin nodes can be displayed in a variety of different ways. Select one of the following or the default,

> **s** - as small circles or spots.
> **c** - in decimal (for $n \leq 32$).

h - in hex.
b - as bits in a 1d or 2d graphic.
B - (for 1 networks only) as bits shown in 2d .

If **b** is selected, the bit pattern will be displayed in 1d or 2d depending on the dimensions of the network. If **B** is selected the 1d network is represented in 2d as a square or rectangular bit pattern, size $i \times j$. Default j (the number of rows) is set automatically as the highest factor of n $\leq \sqrt{n}$ (see also 21.6.4). The following prompt allows the default i to be revised,

now 5x2, reset i-axis: *(values shown are examples)*

Selecting the default layout for learning is a short cut, further layout and presentation options will be skipped. The full range of node display options are described in #20.2).

25.2.3 Mutants for single basins on one screen

For single basins only, enter **M** in section 25.2 to show successive mutants on one screen. An appropriate default layout will be presented. This is useful to show the set of one bit mutants of a rule (see section 28.5.3 and figure 28.2). The basin original basin will be shown by itself in top row.

25.2.4 Basin scale, attractor radius

The first layout prompt (section 25.2) gives the current attractor radius, scaled according to the x-axis. In the example, **(def 6)** means 6% of 1/2 of the screen width. To alter the default basin radius, and the scale of attractor basins, enter a new radius whic can be in decimals. The minimum radius allowed depends on the screen resolution and will correspond to a scale of one pixel. A small scale is somtimes required for basins with very long transients.

25.2.5 Basin start position

To alter the default position of the first basin, enter **return** in section 25.2. The following prompt is presented,

start position, % offset from screen centre
x (def -70): **y (def 50):** *(values shown are examples)*

Enter the new polar coordinates for the first basin, first x, then y. Remember that the screen center is at 0,0 and the edges of the screen are as follows, left x= -100, right x= +100, top y= +100, bottom y= -100. A position outside the screen limits is permitted, allowing, for example, the relevant part of a large basin to be displayed.

25.2.6 Show the field as successive basins

If a basin of attraction field was selected (but not a *range* of n in section 7.1), a further option is offered to display each basin in the field singly as successive single basins at the start position. The following prompt is presented,

show field as successive basins-s, and pause-p:

Enter **s** to show successive basins in the basin of attraction fields without a pause, **p** to pause between each basin. Data on each basin is will be displayed, and **return** will generate the next basin.

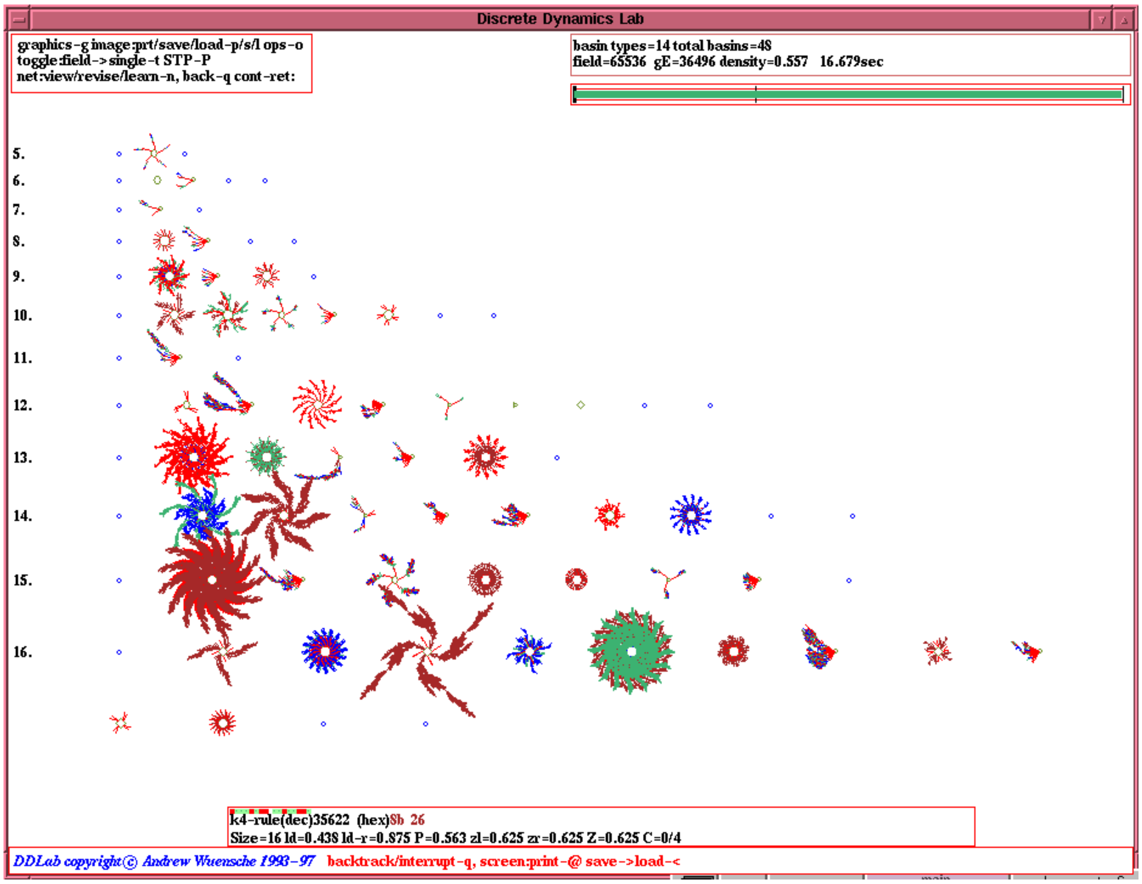


Figure 25.4: The basin of attracion field for a range of network size, n=5-16, shown on the left. The spacing between basins increases at a set rate as larger basins are expected for greater n. In this example, the basin scale is 1.2 (see section 25.2.4), the x, y start position is -80,70 (see section 25.2.5), the initial x, y spacing belween basins is 6,6 (see section 25.2.7), the right border width is 45 (see section 25.2.8), and the x, y spacing increase is set at 12%.

25.2.7 Basin spacing for fields

To alter the default spacing between basins in a basin field (see figure 25.2 left), the following prompt is presented:

> **basin spacing, % of 1/2 screen**
> **x spacing (def 30):** **y spacing(def 30):**
> *(the default values depend on the radius set in section 25.2.4)*

Enter the new spacing according to the polar coordinate scale, first x, then y. Remember that the screen is 200 units wide and 200 units high.

25.2.8 Select minimum right border width

A further option is offered to select the minimum right border width. This is entered as a percentage of the current x-spacing. If the distance between the center of the next basin and the right border is less than this setting, the basin will be drawn at the start of the next row. Negative values for the right border are permitted. The following prompt is presented,

select right border as % of x spacing (default 45.0):
(the default right border depends on the spacing set in section 25.2.7)

25.2.9 Amend the spacing increase for a range of n

If a range of network size n was selected in 7.1, the spacing between successive basins (and fields) is set to increases by a default percentage which may be amended (see figure 25.2 right). The increase is usually required because the size of basins tends to increase with greater n. The following prompt is presented,

select % increase in spacing for successive network size,
x increace (def 10): y increase (def 10): *(values shown are examples)*

Enter the new percentage increase, first for x (for the horizontal spacing), followed by y (for the vertical spacing). This becomes the new default. An example of a range of basin of attraction fields is shown in figure 25.4, and in figure 4.4 in chapter 4.

25.2.10 Accept or revise layout parameters

Finally, the layout preview showing the current settings (as amended) is displayed, with the following prompt to revise or accept,

layout: revise-q, accept-ret:

Enter **q** to revert to the first layout prompt (section 25.2) and re-adjust settings, or **return** to accept the layout and continue.

25.3 Amend the layout on-the-fly

If one of various "pause" options is selected in section 27.2, the angular orientation, spacing and/or position of each successive basin, tree or subtree can be amended as basins are drawn. At the pause, data will be displayed in a top right window, and the following prompt will be presented in a small top centre window,

options-o cont-ret: *(if the subtree pause was set, when each subtree is complete)*
or
options-o layout-a: *(when each basin is complete)*

o gives various options described in section 30.4. **a** gives the following top right prompt,

amend angles-a next pos-p just spacing-ret:

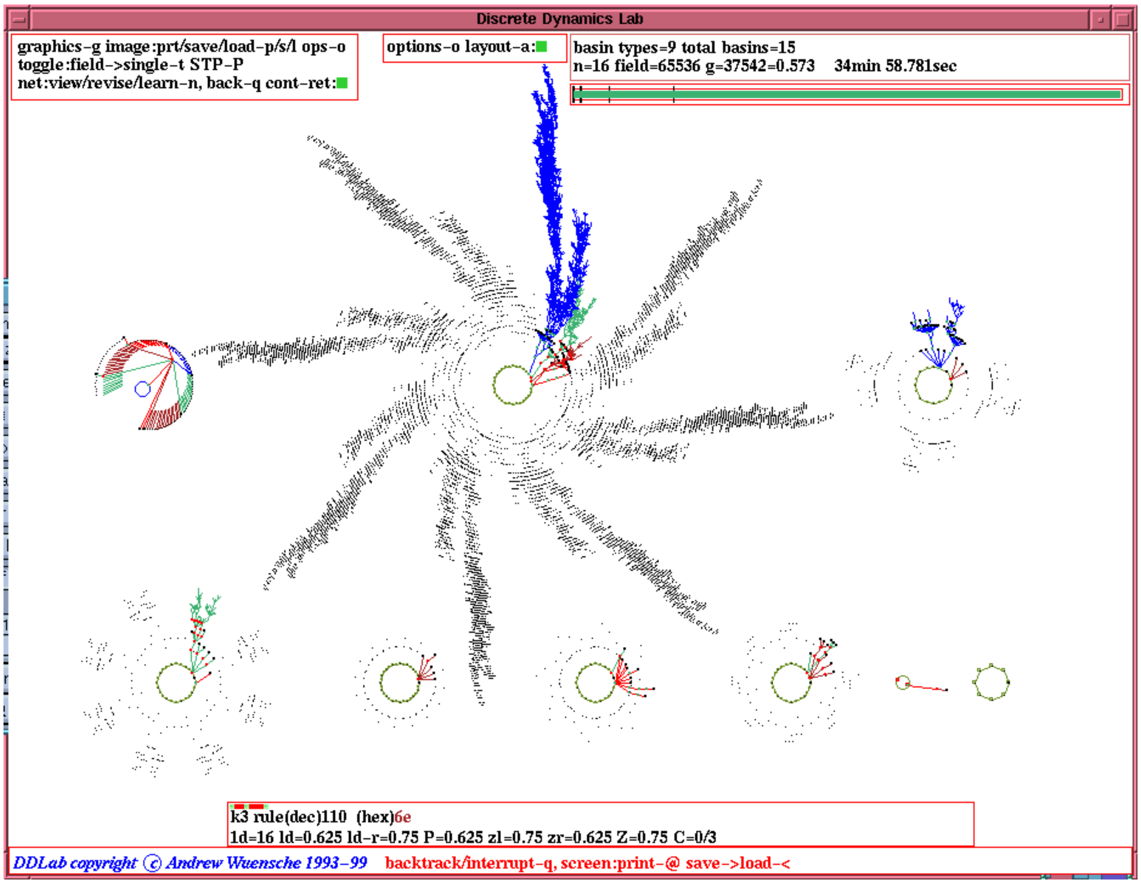


Figure 25.5: Amending the layout of a basin of attraction field on-the-fly, for a CA n=16 rule (dec) 110. The spacing between basins, and the position of the 4th basin, were amended using the options in section 25.3.2 and 25.3.3, to show the basins without overlap. The field is shown with copies of equivalent trees suppressed, exept for garden-of-Eden nodes shown as dots (see section 26.1.2). In this example, the basin scale is 3.8 (see section 25.2.4), the x, y start position is -75,13 (see section 25.2.5), the initial x, y spacing belween basins is 65,73 (see section 25.2.7), and the right border width is 30 (see section 25.2.8).

25.3.1 Amend the orientation and fan angle on-the-fly

Enter **a** in section 25.3 to amend the angular orientation and the spread of in-degree arcs, the "fan angle". Further prompts will be presented in turn,

change basin orientation (now 0), enter angle:
change pre-image fan angle(now 1.00), enter factor: *(values shown are examples)*

See section 26.3 for further details about these settings.

25.3.2 Amend the spacing and right border on-the-fly

If **return** is entered in section 25.3 to amend the spacing (and right border). Prompts are presented as described in section 25.2.7 and 25.2.8 above. Figures 25.5 and 27.1 give examples.

25.3.3 Amend the next position (and spacing) after each basin

Enter **p** in section 25.3 to amend the next position (as well as the spacing). Prompts are presented similar to those described in section 25.2.5 above,

(start= -80,50) amend next pos, % offset from screen centre
x (def -50.2): y (def 50.1): *(values shown are examples)*

The last start position is shown as a reference. Enter the next position's polar co-ordinates, first x, then y. Remember that the screen centre is at 0,0 and the edges of the screen are as follows, left x= -100, right x= +100, top y= +100, bottom y= -100. A position outside the screen limits is permitted. See section refBasin start position for further details about these settings. Figures 25.5 and 27.1 give examples.

After the new position is set the new x co-ordinate becomes the new default x start position for future basins, the default y co-ordinate is not affected.

When this option is complete, the spacing options 25.2.7 and 25.2.8 are presented.

25.3.4 Amend the spacing increase on-the-fly

If a range of network size n was selected in 7.1, the spacing increase between successive basins can be amended as described in section 25.2.9.

Chapter 26

Display of attractor basins

This chapter describes various options to change the default display of attractor basins.

Enter **p** at the first "output parameter" prompt in section 25.2 to jump directly to the display prompts, or reach them by viewing the output parameter in sequence.

For regular 1d or 2d CA, attractor basins can be "compressed" (see section 26.1). In this case the first prompt is in section 26.1.1. Otherwise the first prompt is in section 26.2.

26.1 Compression of equivalent CA dynamics

If the network is a regular 1d or 2d CA (i.e. CA wiring and just one rule), rotational symmetries of the periodic array result in equivalent dynamics [12]. A given network state (periodic pattern) A is embedded in a particular past and future made up of other connected network states. If A is translated around its circle (1d) or torus (2d) by an arbitrary number of moves, and transformed to state B, the states in B's past and future must be identical transformations of the states in A's past and future. The two sets of states are rotational equivalents, and their connection topology is identical.

A complication arises because some states are made up of repeating pattern segments, for example 011,011,011. If the repeating segment size=s, there will be only s rotational equivalent states irrespective of the network size n, whereas with no repeating segments (always the case if n is prime), the number of rotational equivalents is n. This becomes more complicated for a 2d torus were repeating segments exist in two directions, and errors may occur in the algorithm in this case. Compression in 3d has not been implemented.

Note also that states with a given repeating segment size s, cannot be upstream of a state with greater s. In an attractor cycle the value of s for all the states must be equal. The *uniform states* (all 0s and all 1s) with the shortest repeating segment size (s=1) are often powerful attractors.

For regular 1d and 2d CA, "compression" algorithms automatically come into play (unless disabled in section 26.1.1) to compute equivalent basins, transient trees, and subtrees from the uniform states , from each prototype. This considerably speeds up computation. For equivalent basins, only the prototype is shown. The display of equivalent trees and subtrees may also be suppressed to varying degrees.

26.1.1 Suppress compression

The first prompt for regular 1d or 2d CA allows compression (section 26.1) to be suppressed or disabled. The prompt gives a reminder of the CA neighborhood k, size n (or $i \times j$ for 2d), and is presented as follows,

> **DISPLAY: regular 1-d CA, nhood=3, size=10**
> **suppress compression in basins and fields-s:** *(values shown are examples)*

Enter **s** to suppress the compression algorithms.

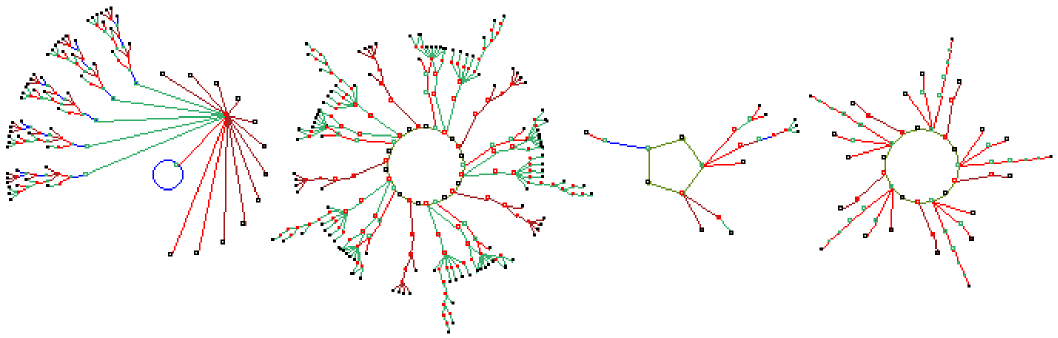


Figure 26.1: A basin of attraction field with compression of rotational equivalent dynamics. k=3, n=10, rule (dec) 110. Only one prototype of each set of equivalent basins is drawn. In addition, equivalent trees on attractor cycles (2nd basin), and subtrees from a uniform states (1st basin) are copied and rotated, thus further speeding up computation as a reverse algorithm need not be re-applied.

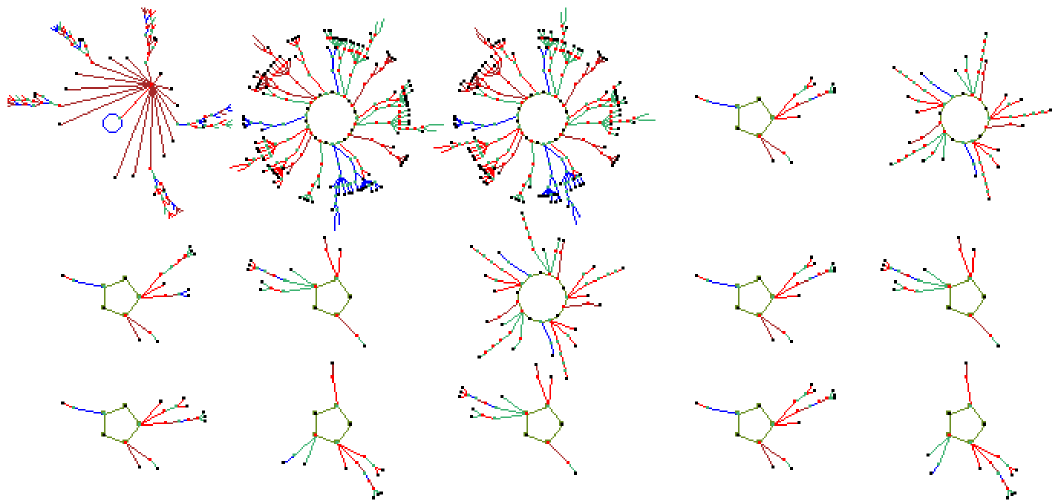


Figure 26.2: The same basin of attraction field as in figure 26.1 without compression of equivalent dynamics, so that all components of the state transition graph are computed from scratch with a reverse algorithm, not copied and rotated as in figure 26.1. All states in state space are represented.

26.1.2 Suppress copies of trees (and subtrees)

If compression remains set, a further option is offered to suppress copies of transient trees from attractor cycles, and subtrees from the two uniform states (all 0s and all 1s). The prototype tree/subtree is drawn in full, whereas the equivalent copies may be suppressed to varying degrees. The following prompt is presented,

> **suppress copies of trees (& subtrees from all 0s, 1s)**
> **all-3, show g-of-E nodes-2, reduce g-of-E nodes-1:**

Enter **3** to suppress all copies of equivalent trees or subtrees. This results in a clearer picture of crowded basins. The other options suppress all but the garden-of-Eden nodes in the equivalent trees or subtrees, showing just a "shadow". Enter **2** for a bold shadow, **1** for a subdued shadow where each garden-of-Eden node is shown as one pixel.

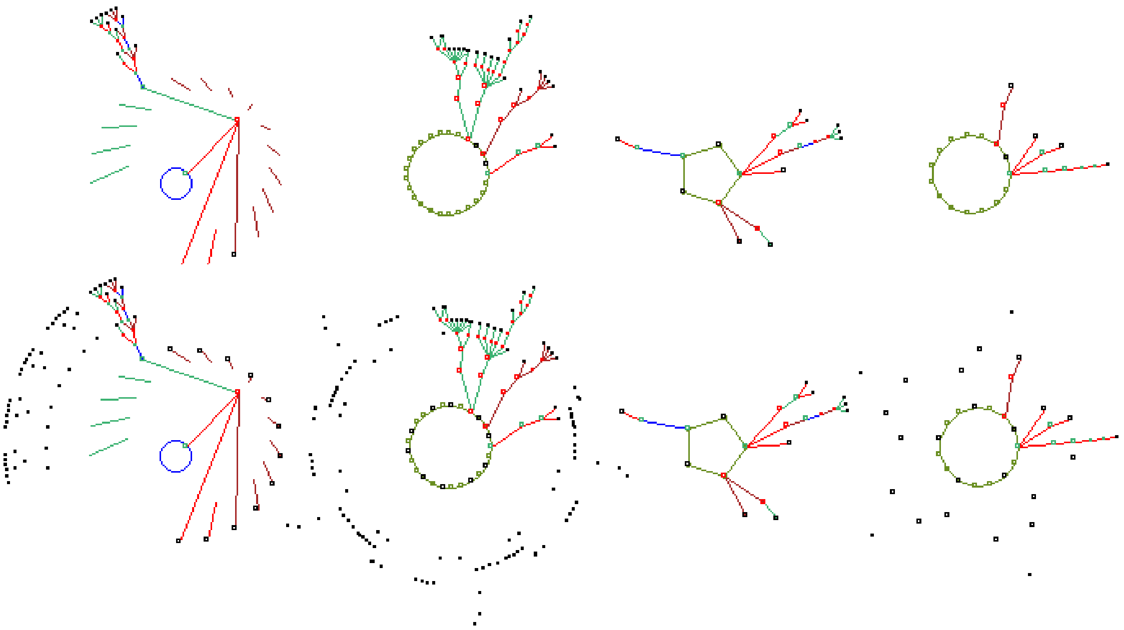


Figure 26.3: Suppressing equivalent subtrees in a a basin of attraction field, k=3, n=10, rule (dec) 110, (as in figure 26.1). *Above*: all equivalent subtrees suppressed, however, short arcs are still shown from a uniform state (1st basin). *Below*: garden-of-Eden states in the suppressed subtrees are shown as spots.

26.2 Node display

The nodes in attractor basins may be displayed in a variety of ways: as small circles or spots, as bit patterns in 1d or 2d depending on the dimensions of the network (though the nodes of a 1d network can also be shown in 2d) or as the hex or decimal value of the pattern, or nodes need not be shown. As well as the primary method of node display, attractor nodes can be independently highlighted as bit patterns. Sets of nodes can also be highlighted in the "learning" routines described in chapter 34. A prompt to set the node display (similar to the following) is presented (see also section 25.2.2). The default depends on previous selections,

> **nodes: 2d-B 1d-b hex-h dec-c spot-s (def-s):**
> **just g-nodes +g, none-n (def-s):** *(***dec-c** *if* $n \leq 32$, **2d-b** *for a 2d network)*

Note that for networks other than regular 1d or 2d CA, the prompts in sections 26.1.1 and 26.1.2 are omitted, and this becomes the first prompt in the display sequence, preceded by the title **DISPLAY**.

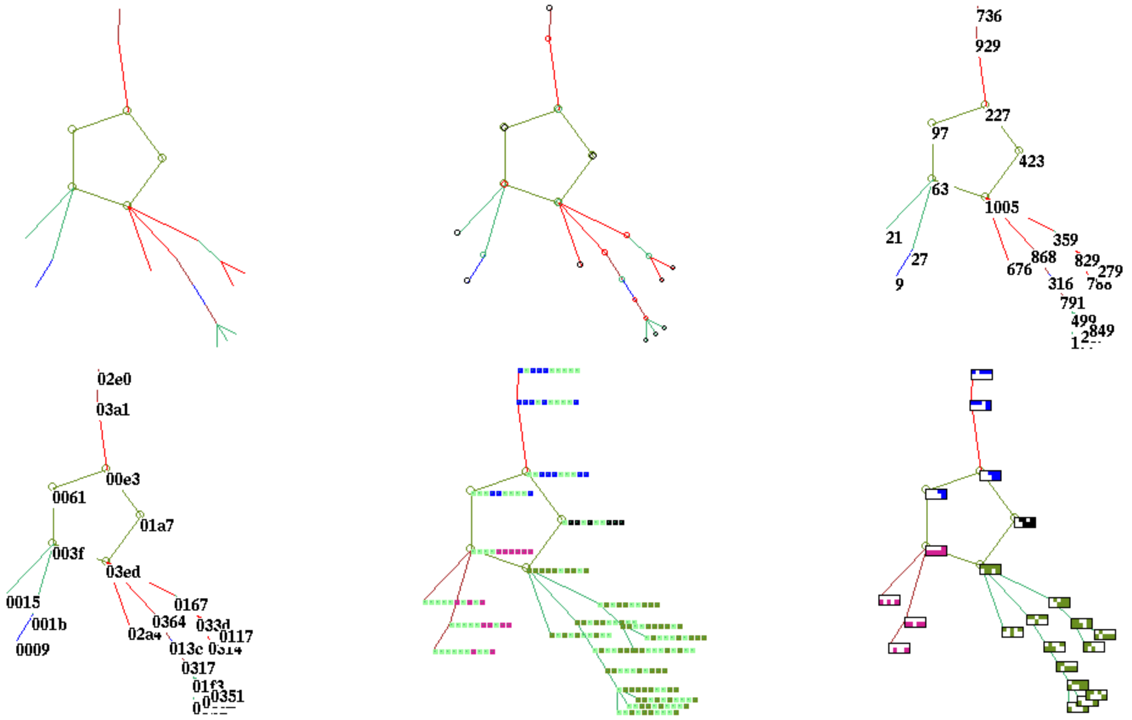


Figure 26.4: Examples of alternative node display. *from the top left*: none (except on the attractor): spots, decimal, hex, 1d bit pattern, and 2d bit pattern, where the $i \times j$ dimensions can be preset. This is the small basin in figures 26.1-26.3, k=3, n=10, rule (dec) 110.

Select the node display as follows,

- **spot-s** ... as small circles or spots. One of 4 colors are set to contrast with transient arc colors (see #18.4), except garden-of-Eden states which are colored white.
- **dec-c** ... in decimal (for the network size, $n \leq 32$).
- **hex-h** ... in hex.
- **1d-b** ... as a bit pattern in 1d for 1d networks.
- **2d-B** ... as a bit pattern in 2d for 1d networks.
- **2d-b** ... as a bit pattern in 2d for 2d and 3d networks.
- **just g-nodes +g** ... enter **g** or one of the options above followedd by **g**, for example **sg** to show the nodes for garden-of-Eden states, and attractor cycles, only.
- **none-n** ... nodes not shown in transients, spots in attractor cycles only.

The colors of spots are set to contrast with transient arcs colors (see section 23.3.1). Decimal and hex numbers are shown black. The colors of 1d and 2d bit patterns are described below.

26.2.1 Bit pattern node colors

2d bit patterns have an outer border, 1d bit patterns do not. The colors of 1s are set to one of 4 colors contrasting with the colors of transition arcs (described in section 23.3.1 where a cycle of four colors is used). Nodes in the same pre-image fan are drawn in the same color (except for the uniform states, see below). If the attractor period is 5 or less, successive fans are assigned a different color so that a given transient tree may have a mix of colors. For attractor periods of 6 or more, all nodes in the same tree are assigned the same color, and the color is changed for the next non-equivalent tree. Note that with compression on (see section 26.1) equivalent trees will be colored identically.

If nodes are highlighted in the learning routine (see #27.1), colors of 1s are black unless highlighted. In general the default node display will correspond to the seed selection method selected in section 21.1.

26.2.2 Bit pattern node colors, uniform states

For a regular 1d or 2d CA, the pre-images of a uniform state's tree is organized into groups of equivalents. These states, and states in their subtrees, are shown in the same color, cycling through four colors.

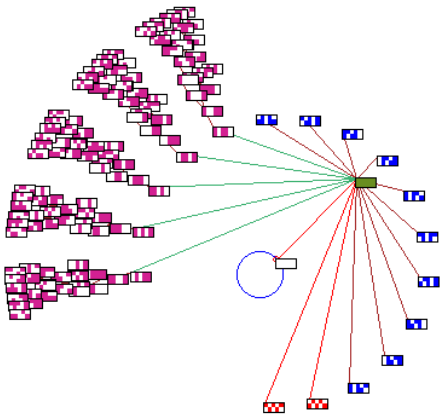


Figure 26.5: The pre-images of a uniform state's tree are organized into groups of equivalents. The bit patterns of these states, and states in their subtrees are shown in the same color, cycling through four colors. This example showed the first basin in figure 26.1, k=3, n=10, rule (dec) 110.

26.2.3 Highlight attractor, or subtree root, as a bit pattern

One or all attractor states, or the root state in a subtree, can be highlighted as a bit pattern, irrespective of the overall node display set in section 26.2. The following prompt is presented,

highlight attractor (or subtree root) as a bit pattern: one-1 all-a:

Enter **1** to highlight one state as a bit pattern in each attractor. The state highlighted is the last attractor state reached in the initial forward run, and is positioned due east on the attractor cycle (see figure 26.10.

Enter **a** to highlight all non-equivalent attractor states for regular 1d or 2d CA, as in figure 26.6, if compression is set in section 26.1.1. Otherwise if **a** is entered all attractor states will be highlighted, as in figure 26.7.

If either **1** or **a** is entered, and running backwards for a subtree is subsequently selected in section 29.1, the subtree root will be highlighted as a bit pattern as in figure 26.9

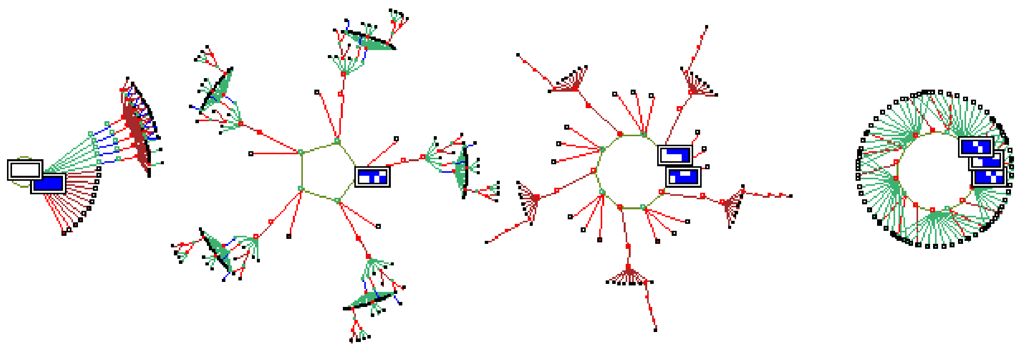


Figure 26.6: Highlighting all non-equivalent attractor states as a 2d bit pattern for the basin of attraction field of a 1d CA, k=3, n=10, rule (dec) 111. In this example other node are shown as spots.

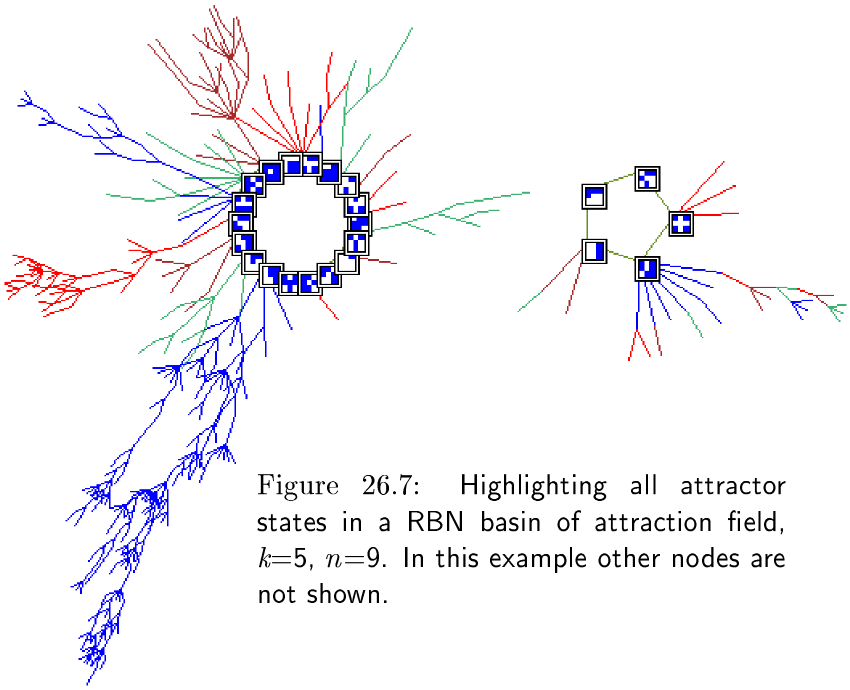


Figure 26.7: Highlighting all attractor states in a RBN basin of attraction field, k=5, n=9. In this example other nodes are not shown.

26.2.4 Show bit pattern nodes for 1d networks in 2d.

If **b** is selected in section 26.2 above, the bit pattern will be displayed in 1d for a 1d network, or 2d for a 2d or 3d network.

For a 1d network, if **B** is selected in section 26.2, or if either **1** or **a** is selected in section 26.2.3, to highlight attractor states, or a subtree root, the 1d network is represented in 2d as a square or rectangular bit pattern, $i \times j$. The default j (the number of rows) is set automatically as the highest factor of n, $\leq \sqrt{n}$ (see also section 21.6.4). The following prompt allows the default i to be revised,

now 7x2=14, reset i-axis: *(example for n=14)*

The 1d network is broken up into successive rows starting with the maximum cell index in the top left hand corner. If the i selected is not a factor of n, some cells will be missing from the bottom row.

26.2.5 Alter decimal or hex node scale

if nodes were set as dec or hex in section 26.2

The default node label size, in decimal or hex, is set automatically depending on the basin scale (see section 25.2.4). This default can be altered by a given factor. The following prompt is presented,

alter default dec/hex node scale, enter factor:

Enter the factor by which the node label size is to be increased or reduced.

26.2.6 Alter node scale, bits

if nodes are set as bits (section 26.2),
or if the attractor or subtree root are to be highlighted as a bit pattern (section 26.2.3)

The default cell scale was set in section 7.3 . This default can be altered by a given factor. The following prompt is presented,

alter cell scale, now 4 pixels: *(value shown is an example)*

Enter the new cell scale in pixels. This new scale will apply to nodes set as bits in section 26.2, and also to attractor or subtree root states, set to be highlighted as a bit pattern in section 26.2.3.

Note that the cell scale is the same as for "backwards" space-time patterns (see section 24.7) and can be changed on-the-fly (see section 30.3).

26.3 Change orientation or fan angle

The next two prompts allow the default orientation of attractor basins (see section 23.2), and of the default pre-image fan angle, to the changed. The following prompts are presented in turn,

change basin orientation (now 0), enter angle:
change pre-image fan angle (now 1.00), enter factor:

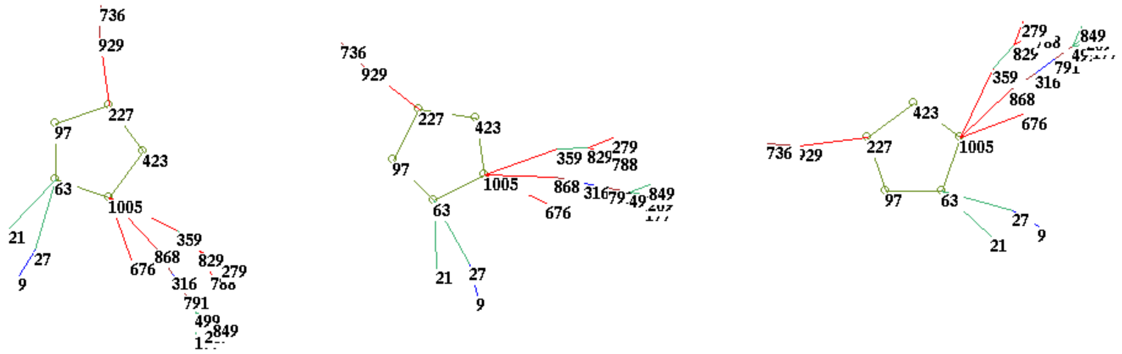


Figure 26.8: Changing the orientation of a single basin of attraction with a decimal seed of 63. This is the small basin in figures 26.1-26.3, k=3, n=10, rule (dec) 110. Nodes are shown in decimal. *left*: orientation=0 degrees. *center*: orientation=45 degrees. *right*: orientation=90 degrees

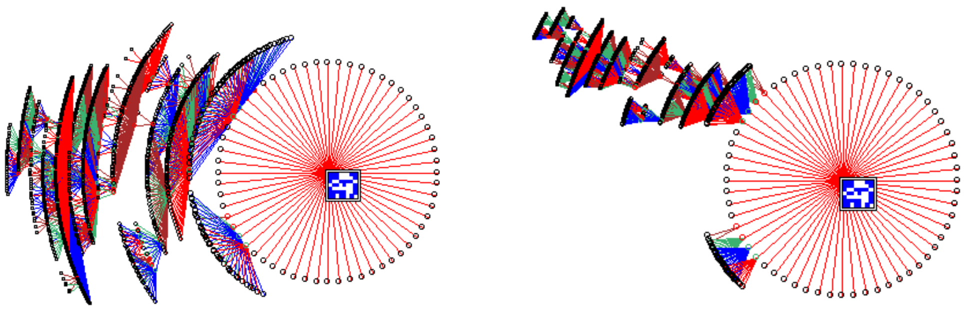


Figure 26.9: Decreasing the pre-image fan-angle of a subtree of a 1d CA, k=3, n=55, rule (dec) 110. The subtree root, (hex) 71f0cf34c0fced, is highlighted as a bit pattern. Other nodes are show as spots. *left*: default fan-angle. *right*: default fan-angle×0.3. Note that the fan converging on the root is always 360 degrees, and is not affected.

26.3.1 Orientation

Changing the orientation allows attractor basins to be rotated anti-clockwise by the angle that is entered in section 26.3. 0 degrees (the default) is due east. Figure 26.8 gives an example. Note that for a single basin of attraction, with the orientation at 0 degrees, the seed state is positioned one step anti-clockwise from east.

Changing the orientation works for subtrees, single basins and basin of attraction fields. The orientation of individual basins in a field can also be change during a pause as fields are drawn (see section 30.2).

26.3.2 Fan-angle

The pre-image fan-angle is the angle containing all the pre-image arcs converging on a node, set automatically in DDLab depending on the number of arcs. It may sometimes be necessary to increase

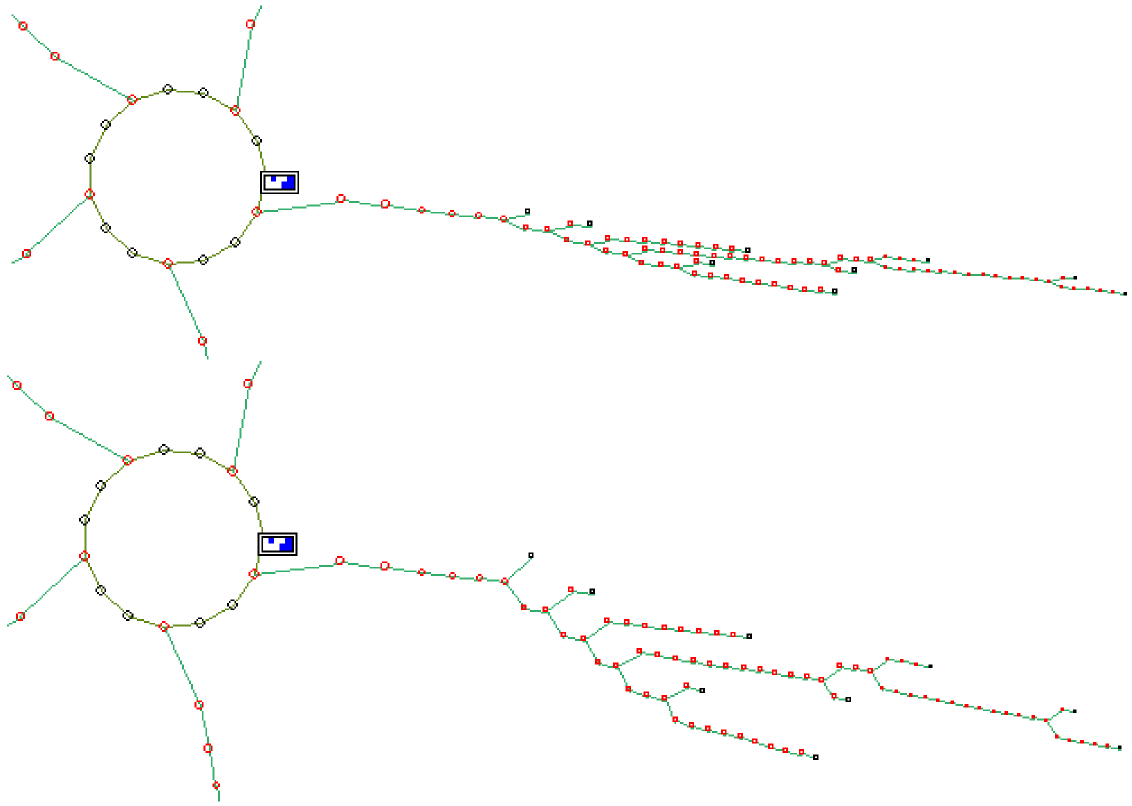


Figure 26.10: Increasing the pre-image fan-angle of a single basin of attraction of a 1d CA, k=3, n=10, rule (dec) 30. Part of the basin including the attractor and a complete tree is shown. One attractor state (due east) is highlighted as a bit pattern. Other nodes are show as spots. This basin has very low in-degree, characteristic of chaos. *left*: default fan-angle. *right*: fan-angle×3, for a clearer picture of subtrees.

the default for chaotic rules with low branching, and decrease the default for denser branching occurring in large networks, or in ordered rules. This can be done by entering a multiplication factor in section 26.3. Figures 26.10 and 26.9 show examples.

Chapter 27

Pausing attractor basins, and data

This chapter describes methods of pausing attractor basins to show data at various levels of detail, and options to print and save this data. A further option, which generates the meta-graph and jump-table of a basin of attraction field, showing where all the 1-bit flips to attractor states end up, is described in section 20.3. The basin of attraction field can then be shown according to the network graph layout (see section 27.8 and 20.7).

To go directly to the **pause/data** category of prompts, Enter **t** at the first "output parameter" prompt in section 25.2, or reach them by viewing the output parameters in sequence.

Data on each field, single basin or subtree, are usually displayed when complete. However, during the drawing of attractor basins, a pause may be set to show intermediate, more detailed, data. This includes data on each tree (or subtree from the two uniform states, all 0s and all 1s), data on each basin in a field, or each field in a range of fields.

A pause also allows the layout of attractor basins to be amended on-the-fly as described in section 25.3.

27.1 Pause after each field for a range of fields

If a basin of attraction field for range of network size was specified in section 7.1, the following prompt is displayed,

PAUSE/DATA pause after each field-f, don't pause-default:

Enter **f** to pause and see the data for each successive field, otherwise only the data in the last field of the sequence will be shown.

27.2 Pause after each basin or tree

The following prompt selects (or deselect) a pause after each basin, or after each tree. This applies in all cases, i.e. for a basin of attraction field, a single basin, or a subtree, including a range set in section 7.1.

PAUSE/DATA pause for data: tree-2, basin-1, none-def:
(the title PAUSE/DATA is show if this is first prompt, i.e. for a single bain or subtree)

Enter **1** to pause and see the data for each successive basin, or **2** to also pause and see data on each successive tree (or subtree).

If a field was selected to be shown as successive basins in section 25.2.6, pausing is the default, and the prompt reads as follows,

pause for data: tree-2, basin-0, basin-def:

Enter **0** not to pause.

27.3 Examples of data

The data are displayed in a top right window. Examples of the types of data displayed, and the decoding of the data, are shown below. More detailed data, including a list of states belonging to the different attractor basins, may also also printed or saved to a file (see sections 27.4 - 27.7).

27.3.1 Data on basin of attraction fields

By default, when a basin of attraction field is complete, unless a range of fields was specified (see sections 7.1 and 27.1), data on the field is presented in a top right window, An example of data on a field is shown below, for k=3 rule (dec) 110, n=12, as shown in figure 27.1,

for a single field, or if pausing in a range of fields
basin types=5 total basins=11
n=12 field=4096 g=1971=0.481 0.641sec

If the basin of attraction field is interrupted (by entering **q** twice, see section 30.2), the data on the incomplete field appears as in the example below,

EARLY EXIT basin types=3 total basins=3
n=12 sspace=4096 field=3358 g=1971=0.481

	decode of data
EARLY EXIT	indicates that the field was interrupted
basin types=	the number of basin prototypes displayed
equiv basins=	the total number of basins in the field
n=	the network size
field=	the total number of states in the field, which should equal the size of state-space, 2^n, unless interrupted If state-space$\neq 2^n$, the size of state-spaces (**sspace=**) will be displayed as well as the incorrect size of the computed field (**field=**). This may indicate an error if the field was not interrupted
g=x=y	the number and density of garden-of-Eden states in the field
x**min** y**sec**	the time taken to draw the field

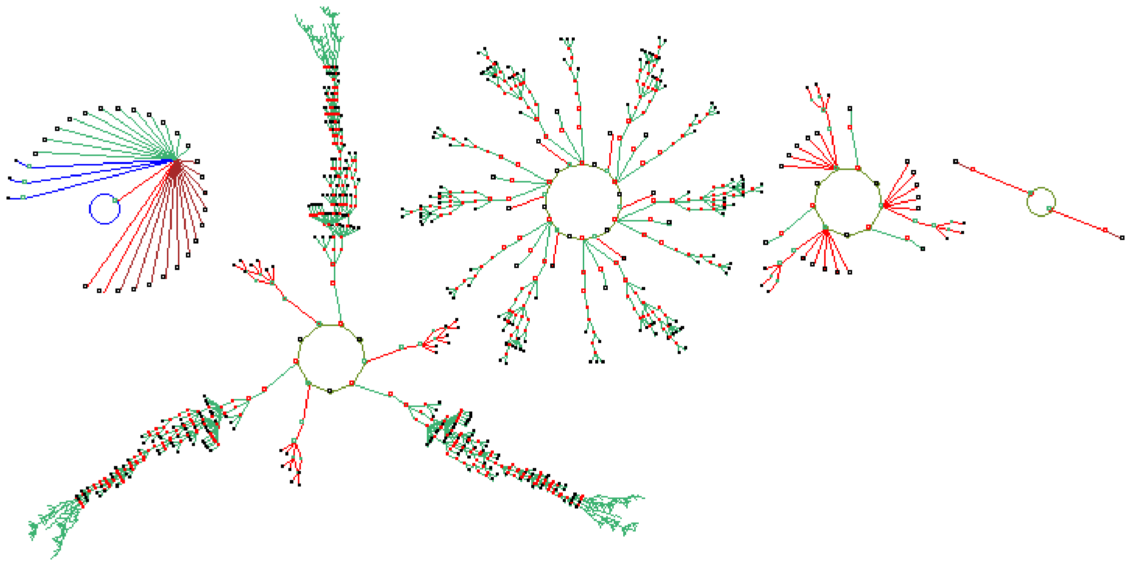


Figure 27.1: A basin of attraction field of a CA, k=3, n=12, rule (dec) 110, relating to various data outputs, displayed in section 27.3, and saved in sections 27.6.1 - 27.6.4. In this example the default display and layout were amended using the options described in sections 25.3 and 26.3.

27.3.2 Errors in basin of attraction fields

Inconsistencies and errors in computing attractor basins can be caused by parameter changes while the basin is in the process of being drawn. One such change is to abandoning a tree and continuing with the next tree described in section 30.2.1. Another is to change some aspect of the network itself, which is possible while the drawing of the attractor basin is interrupted in section 30.2.

If such an error occurs in a basin of attraction field, the progress bar (section 30.1) is likely to go off its scale, the size of state-space as well as the field will be indicated in the data window, for example **sspace=1024 field=1486**. A message will also appear below the progress bar, for example, **ERROR excess states=462** indicating that more states were computed than the size of state-space. 30.2.3 gives further details.

Note that similar errors can occur in single basins or subtrees, but these will not be indicated.

27.3.3 Data on basins

An example of data on a basin is shown below, for k=3 rule (dec) 110, n=12, as shown in figure 27.1,

for a single basin
equiv basins=4 att(hex)=0c 1c
period=9 size=831=81.2% g=402=0.484 ml=35 mp=13 0.579sec
for basin 2 if pausing a field
basin 2, equiv basins=4 att(hex)=00 73
period=9 size=831 81.2% g=402=0.484 ml=35 mp=13

If the basin is interrupted (by entering **q** twice, see section 30.2), the data on the incomplete basin appears as in the example below,

EARLY EXIT equiv basins=4 att(hex)=00 73
period=9 size=42=4.1% g=21=0.5 ml=5 mp=4 1.515sec

	decode of data
EARLY EXIT	indicates that the basin was interrupted
basin x	the basin in question is the prototype of the 2nd type
equiv basins=	number of equivalent basins of this type
att(hex)=	the "first" attractor state, shown in hex
period=	attractor period
size=x	size of the basin,
x**%**	and the percentage of state-space made up by the basin (and its equivalents)
g=x**=**y	the number and density of garden-of-Eden states in the basin
ml=	the maximum number of levels in the basin from the attractor
mp=	the maximum in-degree found in the basin.
x**min** y**sec**	the time taken to draw the basin

27.3.4 Data on trees

An example of data on a tree is shown below, for k=3 rule (dec) 110, n=12, basin 2 (as shown in figure 27.1),

for tree 3 if pausing basin 2
tree=3, no=3 size=263 (tree types=3 att=9)
root(hex)=07 30 g=127=0.483 ml=35 mp=13

	decode of data
tree=	the tree type index
no=	number of trees of this type
size=	size of the tree (including the root)
tree types=	total number of non-equivalent tree types
att=	attractor period
root(hex)=	the state at the root of the sub-tree, in hex.
g=x**=**y	the number and density of garden-of-Eden states in the tree
ml=	maximum number of levels in the tree from the attractor
mp=	maximum in-degree found in the tree

27.3.5 Data on subtrees

If generating just one sub-tree from an arbitrary state, on completion, data will be displayed in a top right window. These examples are for the k=5 rule (hex) aa 55 66 a1.

The first example below is for a large network where n = 150, k=5 rule (hex) aa 55 66 a1, (see figure 27.3.5). A state chosen at random (28c16f6629153200cef8ec6b3a37bd2ff9be9f in hex) was iterated forward by 3 steps, and the subtree was generated from the resultant state reached, the

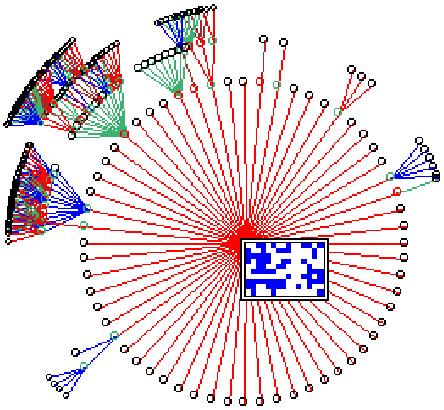


Figure 27.2: A subtree for a CA, $n = 150$, k=5 rule (hex) aa 55 66 a1, illustrating the data in section 27.3.5. The subtree root is shown as a 2d bit pattern (15×10). The default fan angle was reduced by a factor of 0.3 in section 26.3.2.

subtree root. This is necessary because a random state is likely to be a garden-of-Eden state. For such a large network ($n > 56$) the root is not shown. Larger root states are shown when data is printed or saved (see section 27.6.7.

subtree=301
g=255=0.847 ml=4 mp=61 23.119sec

If the subtree is interrupted (by entering **q** twice, see section 30.2), the data on the incomplete subtree appears as in the example below,

EARLY EXIT part subtree=237
g=84=0.354 ml=3 mp=61 9.400sec

	decode of data
EARLY EXIT	indicates that the subtree was interrupted
tree=	the tree type index.
no=	number of trees of this type.
subtree=	size of the subtree (including the root)
part subtree=	size of an interrupted subtree
root(hex)=	the state at the root of the sub-tree, in hex
g=x**=**y	the number and density of garden-of-Eden states in the tree
ml=	maximum number of levels in the tree from the attractor
mp=	maximum in-degree found in the tree.

The second example is for a smaller size, $n = 14$, for the same CA rule (see figure 27.3.5), where a random seed was iterated forward by 33 steps before generating the subtree. However, the state reached (the root of the subtree) was on the attractor, so all the states in the basin were generated. This is indicated by **subtree=basin**. DDLab keeps track of a repeat to prevent the subtree running backwards for ever. The resulting graph is drawn as a subtree (see section 23.3.3 but is made up of all the states in the basin.

subtree=basin=6895 seed(hex)=10 f1
g=2505=0.363 ml=84 mp=6 26.014sec

	decode of data
subtree=	size of the subtree.
subtree=basin=	size of the subtree. subtree=basin indicates the root state is on the attractor, the whole basin is generated.
root(hex)=	the state at the root of the subtree, shown if $n \leq 56$.
g=x**=**y	number of garden-of-Eden states in the subtree.
ml=	maximum number of levels in the subtree from the root.
mp=	maximum in-degree found in the subtree.
x**min** y**sec**	the time taken to generate the subtree.

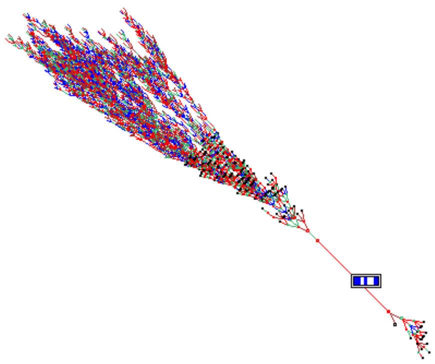


Figure 27.3: A subtree for a CA, $n = 14$, k=5 rule (hex) aa 55 66 a1, illustrating the data in section 27.3.5. The subtree root (shown as a 2d bit pattern) lies on the attractor, so all states in basin are shown.

27.3.6 Data on subtrees from a uniform state

If a subtree pause is selected for a CA in section 27.2 when generating the tree from a uniform state (all 0s or all 1s), the pause will occur after each equivalent set of subtrees has been generated, and data will be displayed on the subtree prototype.

An example of the data on the first prototype subtree is shown below, from a uniform state all 0s, k=3 rule (dec) 96, n=12,

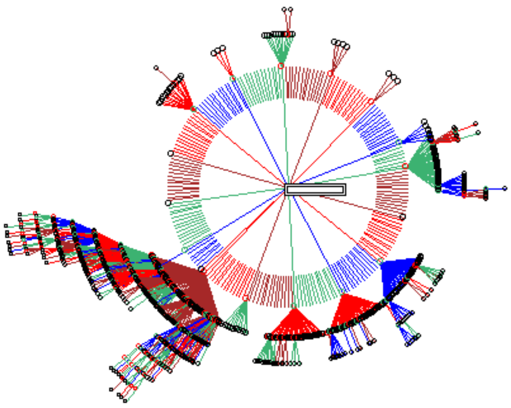


Figure 27.4: Prototype subtrees from a uniform state,all 0s, for a CA, $n = 14$, k=3 rule (dec) 96, illustrating the data in section 27.3.5. Copies of the prototypes have been suppressed in section 26.1.2, and the default fan angle reduced by a factor of 0.3 in section 26.3.2.

seg-fan=98 (11 segs) seg-tree 1, no=12 size=264
seg-root(hex)=00 02 g=199 gd=0.754 ml=8 mp=49

	decode of data
segfan=	the total number of pre-images of the uniform state.
(x seg)	made up of x segments or groups of rotation equivalent states.
seg tree x	the subtree type index.
no=	number of subtrees of this type.
size=	size of the subtree (including the root state).
seg-root(hex)=	the state at the root of the sub-tree, in hex.
g=x=y	the number and density of garden-of-Eden states in the subtree.
ml=	maximum number of levels in the subtree from the attractor.
mp=	maximum in-degree found in the subtree.

Once all prototype subtrees have been generated, data on the complete tree is presented as follows (decode as in section 27.3.5),

subtree=16333 seed(hex)=00 00
g=13364 gd=0.818 ml=9 mp=212 12.250sec

27.4 Print data

As attractor basins are drawn, the data described above (sections 27.3.5 - 27.3.3) may be printed to a printer (for DOS[1]), or to the xterm window (for Unix and Linux) The following prompt is presented,

print data
basin data only-1, and tree data-2:

Enter **1** or **2** to select the level of data to be printed. The format for printing the data (and the decoding of the data) is shown in the examples in section 27.6 below.

27.5 Save data

As attractor basins are drawn, detailed data, including the data described above (sections 27.3.5 - 27.3.3) may be saved to a `.dat` ascii file The following prompt is presented,

save data
basin data only-1, and tree data-2, show states-s or +s:

Enter **1** or **2** to select the type of data to be saved. Alternatively enter **s** to show a list of states with information on where they occur in the field (which basin, which level, number of pre-images).

[1]As just ascii text is printed, the printer limitations described in section 5.5.2 are likely to be less severe

Enter **1s** or **2s** for basin or tree data, plus the list of states and relevant information.

If any of these options are selected, further prompts will appear to set the file name etc. (see Filing, chapter 35) . The format of this ascii data file (and the decoding of the data) is shown in the examples below (section 27.6).

27.6 Format of printed or saved data

Examples of the data format, printed to a printer (DOS) or to the xterm window (UNIX and Linux) in section 27.4, or saved to a `.dat` ascii file in section 27.5 are shown and decoded below.

New data to be saved list should be given a new file name, or the default filename `my_data.dat` should be deleted, as data is appended to the end of the file.

27.6.1 Network parameters file data

If **1** or **2** is selected in section 27.5, the following network parameters are given first, including the neighborhood size, rule number, dimension, size and rule parameters.

This example relates to the figure 27.1.

data *decode*

`k3 rule(dec)110 (hex)6e` ... nneighborhood size k, rule in decimal (for $k \leq 5$) and hex.
`1d=12 ld=0.625 ld-r=0.75 P=0.625` ... network dimension,size, λ and λ_{ratio} rule parameters.
`zl=0.75 zr=0.625 Z=0.75 C=0/3` ... Z_{left}, Z_{right} and the Z-parameter, canalizing inputs.

This information is only given for single rule networks. For mixed rule (and mixed k) networks, the complete network parameters can be printed and saved (also to a `.dat` ascii file) as described in section 19.5.

27.6.2 Basin field file data

An example of the complete data for the basin of attraction field in figure 27.1 (enter **1** in section 27.5),

data *decode*

`k3 rule(dec)110 (hex)6e` ...
`1d=12 ld=0.625 ld-r=0.75 P=0.625` ...
`zl=0.75 zr=0.625 Z=0.75 C=0/3` ...network parameters, described in section 27.6.1 above
`ty. at no (p) s % g gd ml mp` ... reminder of data order, key in section 27.6.3 below
`1. 0000 1 (1) 34 0.83% 29 0.853 3 29` ... basin 1 data
`2. 0073 4 (9) 831 81.2% 402 0.484 35 13` ... basin 2 data
`3. 07c7 2 (18) 312 15.2% 138 0.442 10 6` ... basin 3 data
`4. 01c7 2 (9) 51 2.49% 27 0.529 4 7` ... basin 4 data
`5. 0777 2 (2) 6 0.293% 2 0.333 2 2` ... basin 5 data
`basin types=5 total basins=11` ... basin summary
`n=12 field=4096` ... network size, field size
`g=1971=0.481` ... total garden-of-Eden states, and density

27.6.3 Key to basin data order

The key to data on basins is set out below, where the labels `ty. at no (p) s % g gd ml mp` are shown as a reminder, If CA compression is suppressed, (see section 26.1), or for a non-CA network, there are no equivalent basins, data on every basin will be shown.

data type	*decode*
`ty.` ...	basin type numbered in the order drawn.
`at` ...	the "last" attractor state in hex.
`no` ...	number of equivalent basins of this type (always 1 if compression suppressed).
`(p)` ...	attractor period.
`s` ...	total states in basin.
`%` ...	percentage of state-space made up of this basin type.
`g` ...	number of garden-of-Eden states in the basin.
`gd` ...	the density of garden-of-Eden states in the basin.
`ml` ...	the maximum number of tree levels in the basin from the attractor.
`mp` ...	the maximum in-degree found in the basin.

27.6.4 Tree file data

An example of the same data as for the basin of attraction field in section 27.6.2 above (see figure 27.1), but also including tree (and subtree) data, which precedes the basin data summary (enter **2** in section 27.5). For CA, subtree data is given for each subtree type rooted on the pre-images of a uniform attractor state (all 0s or 1s) (see section 27.3.6).

data *decode*

```
k3 rule(dec)110 (hex)6e ...
1d=12 ld=0.625 ld-r=0.75 P=0.625 ...
zl=0.75 zr=0.625 Z=0.75 C=0/3 ...network parameters, described in section 27.6.1
ty.  at no (p) s % g gd ml mp ... reminder of data order, key in section 27.6.3 above

tree.  root no s g gd ml mp ... reminder of tree data order, key in section 27.6.5 below
seg-fan=29 (4 segs) ...uniform state in-degree, and number of sub-tree types
1.  0aaa 2 1 1 1 2 0 ...data for each subtree type, in basin 1
2.  0ab6 12 1 1 1 2 0
3.  0ad6 12 1 1 1 2 0
4.  0db6 3 2 1 0.5 3 1
1.  0000 1 (1) 34 0.83% 29 0.853 3 29 ...data, basin type 1
1.  0fd1 3 13 7 0.538 5 4 ...data for each tree type in basin 2
2.  0d70 3 1 0 0 0 0
3.  0730 3 263 127 0.483 35 13
2.  0073 4 (9) 831 81.2% 402 0.484 35 13 ...data, basin type 2
1.  037d 6 2 1 0.5 1 2 ...data for each tree type in basin 3
2.  0137 6 1 0 0 0 0
3.  0f1d 6 49 22 0.449 10 6
3.  07c7 2 (18) 312 15.2% 138 0.442 10 6 ...data, basin type 3
1.  0f7d 3 13 8 0.615 4 7 ...data for each tree type in basin 4
2.  0d34 3 1 0 0 0 0
3.  071c 3 3 1 0.333 2 2
4.  01c7 2 (9) 51 2.49% 27 0.529 4 7 ...data, basin type 4
1.  0ddd 2 3 1 0.333 2 2 ...data for each tree type in basin 5
5.  0777 2 (2) 6 0.293% 2 0.333 2 2 ...data, basin type 5
basin types=5 total basins=11 ... basin summary
n=12 field=4096 ... network size, field size
g=1971=0.481 ... total garden-of-Eden states, and density
```

27.6.5 Key to tree data order

Section 27.6.3 gave the the key to data on basins. The key to data on trees is set out below, where the labels `tree. root no s g gd ml mp` are shown as a reminder. Note that for CA, with compression active (see section 26.1), data is also given on subtrees from a uniform state, all 0s or 1s. If CA compression is suppressed, or for a non-CA network, there are no equivalent trees, data on every tree will be shown.

data type	*decode*
`tree.` ...	tree (or subtree) type numbered in the order drawn.
`root` ...	the state at the root of the tree, in hex.
`no` ...	number of equivalent trees of this type (always 1 if compression suppressed).
`s` ...	the size of the tree including the root.
`g` ...	number of garden-of Eden states in the tree.
`gd` ...	the density of garden-of-Eden states in the tree.
`ml` ...	the maximum number of levels in the tree from the root.
`mp` ...	the maximum in-degree found in the tree.

27.6.6 Single basin file data

An example of the data for a single basin, with tree data, is given below (enter **2** in section 27.5). This is the second basin in figure 27.1.

data	*decode*
`k3 rule(dec)110 (hex)6e` ...	
`1d=12 ld=0.625 ld-r=0.75 P=0.625` ...	
`zl=0.75 zr=0.625 Z=0.75 C=0/3` ...	network parameters, described in section 27.6.1
`1.  0fd1 3 13 7 0.538 5 4` ...	data for each tree type
`2.  0d70 3 1 0 0 0 0`	
`3.  0730 3 263 127 0.483 35 13`	
`single basin.  0073 4 (9) 831 81.2% 402 0.484 35 13` ...	data for basin

The tree data order is as shown above in section 27.6.5, and the basin data order is as shown in section 27.6.3, but starting with the the "last" attractor state in hex. The reminders of the data order are not shown.

27.6.7 Subtree file data

An example of the data for the subtree (n=150) shown in in section 27.3.5, and in figure 27.3.5, is given below (enter **2** in section 27.5).

data	*decode*
`k5 rule(dec)2857723553 (hex)aa5566a1` ...	
`1d=150 ld=0.469 ld-r=0.938 P=0.531` ...	
`zl=0.938 zr=0.672 Z=0.9375 C=0/5` ...	network parameters, described in section 27.6.1
`subtree=301 root(hex)=0d26c9c2f8057320cc23e07009bfe1f3` ...	subtree size, root in hex
`ca2201` ...	the root in hex continues, if more than 32 hex characters
`g=255=0.847 ml=4 mp=61` ...	more data (see key, section 27.6.5

27.7 List of states

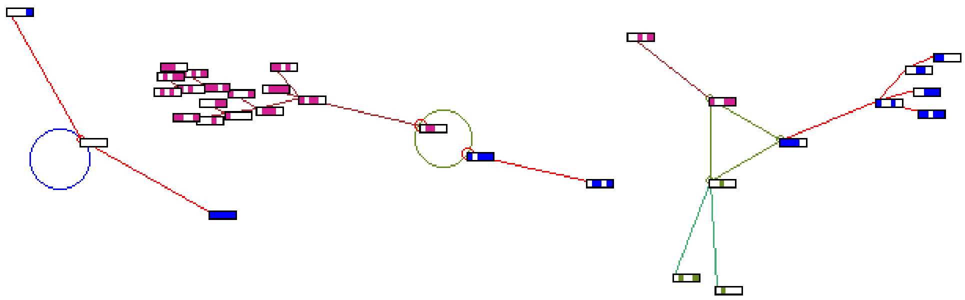


Figure 27.5: The basin of attraction field of an RBN, with states listed in section 27.7. n=5, with mixed wiring k=3, and a homogenious rule (dec) 110.

Examples of the list of states and data (with explanatory notes) are shown in sections 27.7.1 to 27.7.3 below, for a basin of attraction field of an RBN[2] (see figure 27.5).

The network is as follows: n=5, with mixed wiring k=3, and a homogenious rule (dec) 110. The wiring shown as a matrix (see section 17.2.1) is as follows,

```
     2.1.0.

4..  4 1 2
3..  3 4 4
2..  0 3 4
1..  2 2 4
0..  1 2 2
```

As well as showing just the list of states and data, the list can also be broken up into sections giving basin data, or in addition tree data, by entering **1s** or **2s** in section 27.5. Both of these options show network parameters, a reminder of the basin data (and tree data) order at the top, and a summary of data for the basin of attraction field at the bottom, as in section 27.6. Compare these lists with the basin of attraction field shown in figure 27.5.

27.7.1 Just the list of states

The example below shows just the list of states and data (and explanatory notes), enter **s** in section 27.5, The first column is the state bit-string. The next three columms indicate,

the basin number As drawn in sequence. For a CA, if compression has not been suppressed (see section 26.1, this is the basin prototype number. For a subtree or single basin the number is always 1.

level How many steps (levels) away from the attractor for a basin of attraction, or from the subtree root for a subtree. If this is 0 then the state is on the attractor, or is the subtree root.

pre-images The number of the state's immediate predecessors (pre-images). If this is 0 then the state is "garden-of-Eden".

[2]Note that in an RBN compression does not apply (see section 26.1).

data *explanatory notes*

```
00000 1 0 3
11111 1 1 0
00001 1 1 0
10111 2 0 2
01101 2 1 0
01100 2 0 2
10110 2 1 3
11010 2 2 0
01111 2 2 0
01110 2 2 2
10001 2 3 1
10000 2 3 2
11101 2 4 2
00011 2 4 0
00010 2 4 1
10101 2 5 1
10100 2 5 2
11001 2 5 0
11100 2 6 0
01011 2 6 0
01010 2 6 0
11110 3 0 2
10010 3 1 3
11011 3 2 0
00111 3 2 0
00110 3 2 1
11000 3 3 0
10011 3 0 2
00101 3 1 0
00100 3 0 3
01001 3 1 0
01000 3 1 0
   \   \ \ \...the number of predecessors of the state, garden-of-Eden=0
    \   \ \...level, attractor state=0, subtree root=0
     \   \...basin number, subtree or single basin=1
      \.....state shown as a bitstring
```

27.7.2 The list of states with basin data

This example shows the same list as in section 27.7.1 together with basin data (and explanatory notes), enter **1s** in section 27.5,

data *explanatory notes*

```
k3 rule(dec)110 (hex)6e
1d=5 ld=0.625 ld-r=0.75 P=0.625
zl=0.75 zr=0.625 Z=0.75 C=0/3             ...  network parameters
ty.  at  no  (p)  s  %  g  gd  ml  mp    ...  reminder of basin data order
00000 1 0 3                               ...  list of states and data for basin 1
11111 1 1 0
00001 1 1 0
1. 00 1 (1) 3 9.38% 2 0.667 1 3           ...  basin data for basin 1
10111 2 0 2                               ...  list of states and data for basin 2
  ...                                          (complete list not shown)

01010 2 6 0
2. 0c 1 (2) 18 56.2% 8 0.444 6 3          ...  basin data for basin 2
11110 3 0 2                               ...  list of states and data for basin 3
  ...                                          (complete list not shown)

01000 3 1 0
3. 04 1 (3) 11 34.4% 6 0.545 3 3          ...  basin data for basin 3
basin types=3 total basins=3              ...  summary for the basin of attraction field
gE=16=0.5
```

The format and keys of network parameters and basin data are described in sections 27.6.1 - 27.6.3.

27.7.3 The list of states with basin *and* tree data

This example shows the same list as in sections 27.7.1 and 27.7.2 together with basin data *and* tree data (and explanatory notes), enter **2s** in section 27.5, *data* *explanatory notes*

```
k3 rule(dec)110 (hex)6e
1d=5 ld=0.625 ld-r=0.75 P=0.625
zl=0.75 zr=0.625 Z=0.75 C=0/3              ...network parameters
ty.  at  no  (p)  s  %  g  gd  ml  mp      ...reminder of basin data order
   tree. root  no  s  g  gd  ml  mp        ...reminder of tree data order
00000 1 0 3                                ...list of states and data for tree 1, basin 1
11111 1 1 0
00001 1 1 0
   1. 00 1 3  2 0.667 1 3                  ...tree data for tree 1, basin 1
1. 00 1 (1) 3 9.38% 2 0.667 1 3            ...basin data for basin 1
10111 2 0 2                                ...list of states and data for tree 1, basin 2
01101 2 1 0
   1. 17 1 2  1 0.5 1 2                    ...tree data for tree 1, basin 2
01100 2 0 2                                ...list of states and data for tree 2, basin 2
  ...                                         (complete list not shown)
01010 2 6 0
   2. 0c 1 16  7 0.438 6 3                 ...tree data for tree 2, basin 2
2. 0c 1 (2) 18 56.2% 8 0.444 6 3           ...basin data for basin 2
11110 3 0 2                                ...list of states and data for tree 1, basin 3
  ...                                         (complete list not shown)
11000 3 3 0
   1. 1e 1 6  3 0.5 3 3                    ...tree data for tree 1, basin 3
10011 3 0 2                                ...list of states and data for tree 2, basin 3
00101 3 1 0
   2. 13 1 2  1 0.5 1 2                    ...tree data for tree 2, basin 3
00100 3 0 3                                ...list of states and data for tree 3, basin 3
01001 3 1 0
01000 3 1 0
   3. 04 1 3  2 0.667 1 3                  ...tree data for tree 3, basin 3
3. 04 1 (3) 11 34.4% 6 0.545 3 3           ...basin data for basin 3
basin types=3 total basins=3               ...summary for the basin of attraction field
gE=16=0.5
```

The format and keys of network parameters, basin data and tree data are described in sections 27.6.1 - 27.6.5.

27.8 The attractor meta-graph

The attractor meta-graph, and the "jump-table", represent the probabilities of jumping between basins due to 1-bit purturbations to attractor states, and gives some insight into the stabitity and mobility in attractor space. For a fuller description see section 20.3. For basin of attraction fields only, the following prompt is presented to select the attractor-meta graph,

attractor meta-graph -m:

Enter **m** to show the meta-graph. For CA, if "compression" is active it will be turned off (see section 26.1), and the following message will be displayed,

... compression in basin/fields turned OFF.

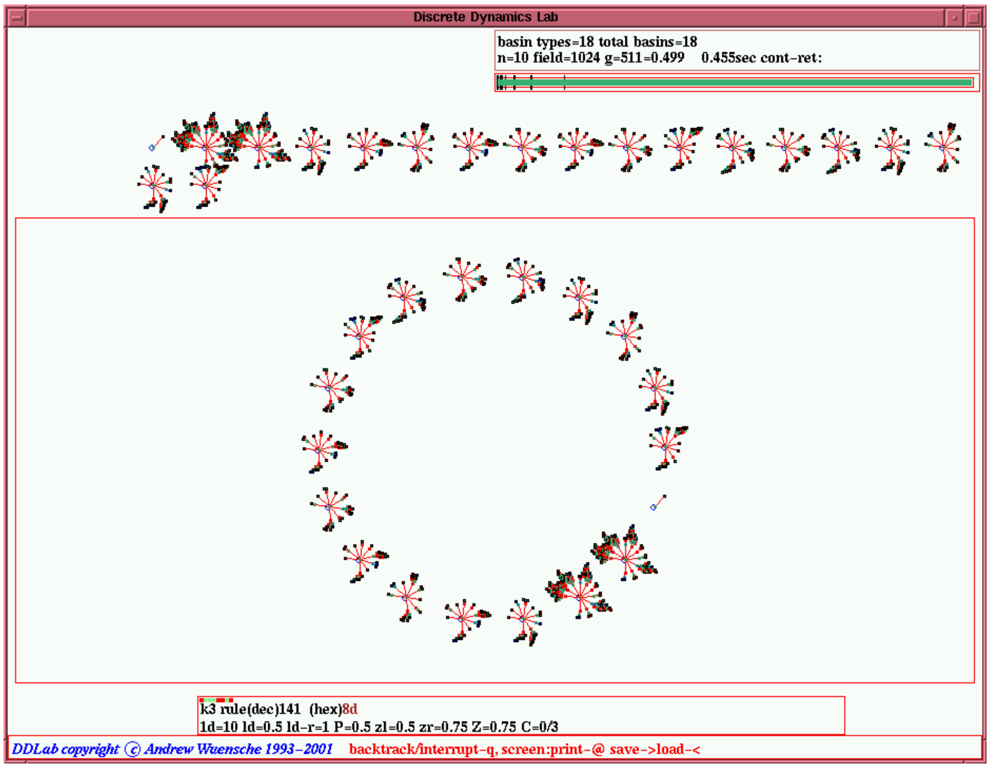
If **m** is selected, a further prompt is presented to display the jump table in the xterm window (not for DOS). The table can also be displayed in the meta-graph window within DDLab (see section 20.10).

show jump-table: numbers-n fractions-f both-b: *(not for DOS)*

Enter **n** for the number of jumps, **f** for the fraction of jumps, or **b** for both.

If the meta-graph option is selected above the basin of attraction field will be generated first, then DDLab automatically generates the meta-graph. The meta-graph can also be selected in section 31.18.2 for the attractor histogram.

The meta-graph has a number of default layouts, circle, spiral, 1d, 2d, 3d, and many options for rearanging its layout, described in 20.3.1. A further option (section 20.7) allows the basin of attraction field to be redraw within the attractor meta-graph window, according to the current graph layout, providing many more layout possibilities to those described in chapter 25. Figure 27.7.3 gives examples.



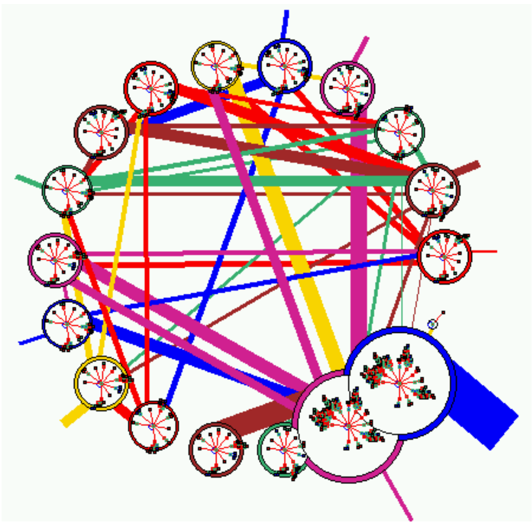


Figure 27.6: *above*: The basin of attraction field redraw within the attractor meta-graph window, according to the current graph layout, as well as in the normal layout at the top, for a 1d CA, n=10, rule 141. *left*: as above, but showing the meta-graph itself, with the basins redrawn inside the meta-graph nodes, which are scale according to basin volume.

Chapter 28

Mutation of attractor basins

When the attractor basin have been drawn for a given network, a new attractor basin may be immediately generated with the same presentation settings, but with a change, or mutation, to some aspect of the network. This process can be repeated indefinitely and automatically. Mutations are either cumulative or relate back to the starting network.

Mutations may comprise just one bit in a rule table or just one wire move in a RBN. A whole spectrum of mutations can be preset from small to large, or a random new network can be selected. This chapter describes the various mutation options.

Mutant basins can be grouped on one screen screen by selecting this option in section 25.2.3. This is especially useful to show the set of one bit mutants of a rule (see section 28.5.3).

28.1 The first mutation prompt

If **m** is selected at the first "output parameter" prompt in section 24.1, or if the output parameter prompts are viewed in sequence, the first mutation prompt is presented in a top right window as follows,

> **MUTATION for next: none-n chain-c**
> **wiring-w rule: special-s, set bits-b, single bit (def):**

Enter **return** to mutate a CA rule by flipping one bit at random. For a mixed rule network, a random bit will be flipped in the rule of just one cell chosen at random. Successive mutations are cumulative.

Enter **n** to suppress mutations. This may be useful to regenerate the attractor basin for the same network, but with a different layout or presentation. Other mutation options are described below.

28.2 Chain rules

Enter **c** at the first mutation prompt above (section 28.1) for a random rule chosen from the set of chain rules. These are maximally chaotic rules where the Z parameter equals 1 in one direction only. The rules result in attractor basins with very low in-degree. For larger 1d CA typically a state has just one predecessor, suitable for encryption.

28.3 Mutate wiring

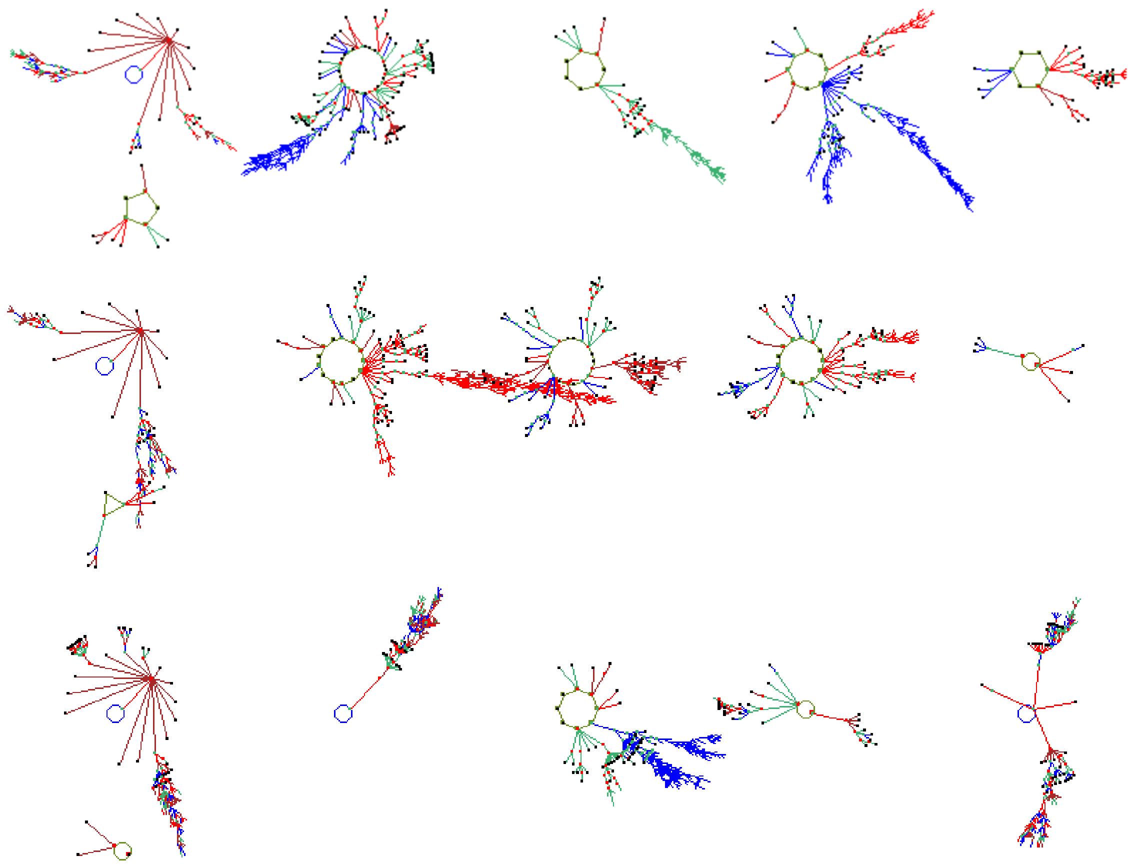


Figure 28.1: Mutation by cumulatively moving one wire at random. The basin of attraction field, k=3, n=10, rule (dec) 110. Starting with CA wiring (resulting in the basin field shown in figure 26.2 with compression suppressed), three mutant basin fields are shown where 1 randomly chosen wire has been moved cumilativly for each mutation.

Enter **w** at the first mutation prompt (section 28.1) to move one or more wires in the network to random positions. A top right prompt similar to the following is presented,

> **relocal-l, wires to move=1/50=2%: all-a number-n percentage-p:**
> *(values shown are examples)*

In this example, n=10, k=5, so the total number of wires is 50. In a mixed k network the total number of wires equals the sum of the n neighborhoods.

The default is to move just one wire. The wire mutations are cumulative. The wire moves are made by picking a random wire of a random cell for each wire move. Note that a regular CA will be re-assigned as a randomly wired (non-local) network if wiring mutation is chosen.

Enter **l** to reset the wiring as local, as in a 1d CA.

Enter **a** for new random wiring for the whole network at each mutation.

28.3.1 Set the number of wires to move

If **n** is entered in section 28.3 above, the following further prompt is presented,

number of wires:

Enter the number of wires to be moved at each mutation.

28.3.2 Set the percentage of wires to move

If **p** is entered in section 28.3, the following further prompt is presented:

percentage of wires:

Enter the percentage of wires to move at each mutation.

28.4 Flip bits

Enter **b** at the first mutation prompt (section 28.1), to alter the default number of bits to be flipped at random. The original default is to flip just one bit. Successive mutations are cumulative. The following option depends on whether the network has one homogeneous rule, or a rule mix. In this example, the n=10 and k=5.

In a network with one homogeneous rule the random bit-flips apply to one rule only. The maximum number of bits that may be flipped equals the size of the rule-table. The following top right prompt is presented,

bits to flip=1/32=12%: all-a number-n percentage-p *(values shown are examples)*

In a network with mixed rules (or mixed k), the total number of bits specifying the rule scheme is the sum of the bits in n rule tables, $n \times 2^k$. This is shown in the prompt below for a k=5, n=10 network, and is the basis for the percentage setting. Mutations are made by picking a random bit of a random cell's rule-table for each bit flip.

bits to flip=1/320=0.312%: all-a number-n percentage-p

Enter **a** to assign an entirely new rule or rule scheme is assigned at each mutation.

28.4.1 Alter the number of bits to flip

If **n** is entered in section 28.4 above, the following further top right prompt is presented,

number of bits:

Enter the new number of bits to flip, which becomes the default.

28.4.2 Alter the percentage of bits to flip

If **p** is entered in section 28.4 above, the following further top right prompt is presented,

percentage of total bits:

Enter the new percentage of bits to flip. The resultant number becomes the default.

28.5 Special rule mutation

Enter **s** at the first mutation prompt (section 28.1) for other special rule mutation options described below. A top right prompt similar to the following is presented,

specify rule mutation
codeno-5 randcode-4 ruleno-3 randrule-def *(***5**,**4** *for totalistic only,* **3** *for* $k \leq 5$ *only)*
bitflip from left: cumulative-1 single-0: *(not for a neighborhood mix)*

28.5.1 Mutate by the rule or code number

If the rule was specified as a totalistic in section 12.2, enter **5** in section 28.5 to decrease the totalistic code number by 1 on each mutation. Enter **4** for a random totalistic code.

For $k \leq 5$, enter **3** in section 28.5 to decrease the rule number by 1 on each mutation. Enter **return** for a random rule.

For a single rule network, when the rule or code number reaches 1 the mutation sequence stops.

For a network with a rule or code mix (or mixed k), each rule or code number in the network is decreased by 1. After reaching 0 the rule or code number restarts at its maximum value.

28.5.2 Assign a random rule or code number

If the rule was specified as a totalistic in section 12.2, enter **4** in section 28.5 for a random totalistic code at each mutation. Otherwise enter **return** for a random rule.

28.5.3 Bit-flip in sequence

If **1** or **0** is entered in section 28.5, at each mutation, successive bits in the rule-table will be flipped, in sequence from left to right. The difference is that for **cumulative-1**, the bits flipped previously will remain unchanged, whereas for **single-0**, the bit flipped previously will be restored to its former state. The "single" bit-flip results in a set of rules 1 Hamming distance from the start rule, the cumulative bit-flip results in a steady increase in the Hamming distance. For a rule mix, the mutation will apply to each rule. Note that finer mutations can be made to a rule if it is first transformed to an equivalent rule with greater k, as described in section 18.9. Figure 28.3 and 28.2 give an examples.

For single basins only, the mutant basins can be grouped on one screen with the original basin by itself in top row, by selecting this option in section 25.2.3. For a bit-flip in sequence, if the set of mutants fits on one screen the program will pause once the set is complete (or when the screen is full). Figure 28.2 gives an example.

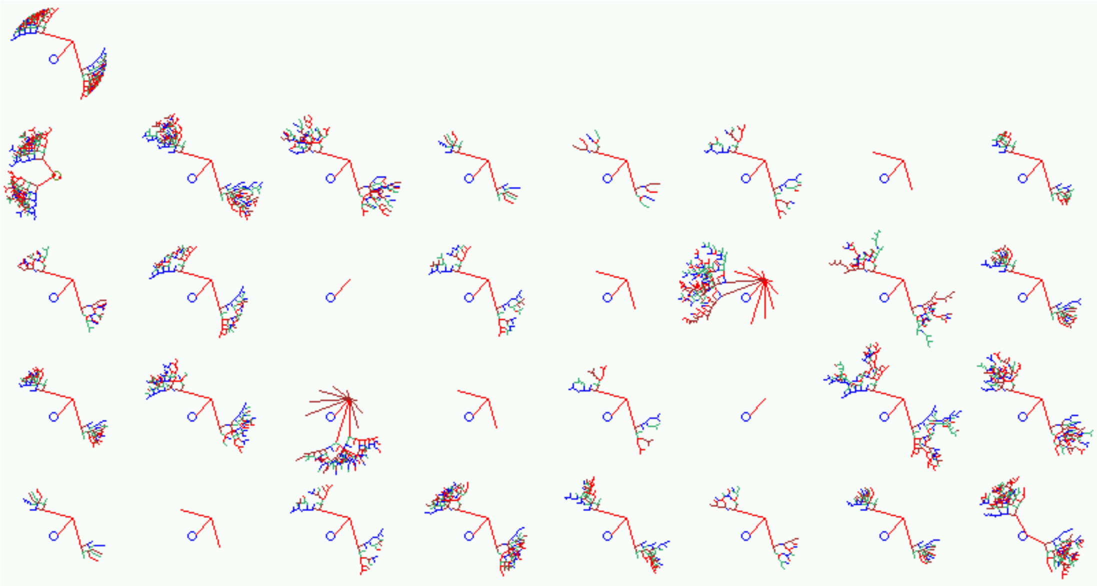


Figure 28.2: 32 mutant basins of attraction shown on one screen. k=*3* rule 195, n=8, seed all 0s. *top row*: The original rule, where all states fall into just one very regular basin. The rule was first transformed to the equivalent k=*5* rule f0 0f f0 0f, with 32 bits in its rule-table. All 32 one bit mutant basin are shown. If the rule is the genotype, the basin of attraction can be seen as the phenotype.

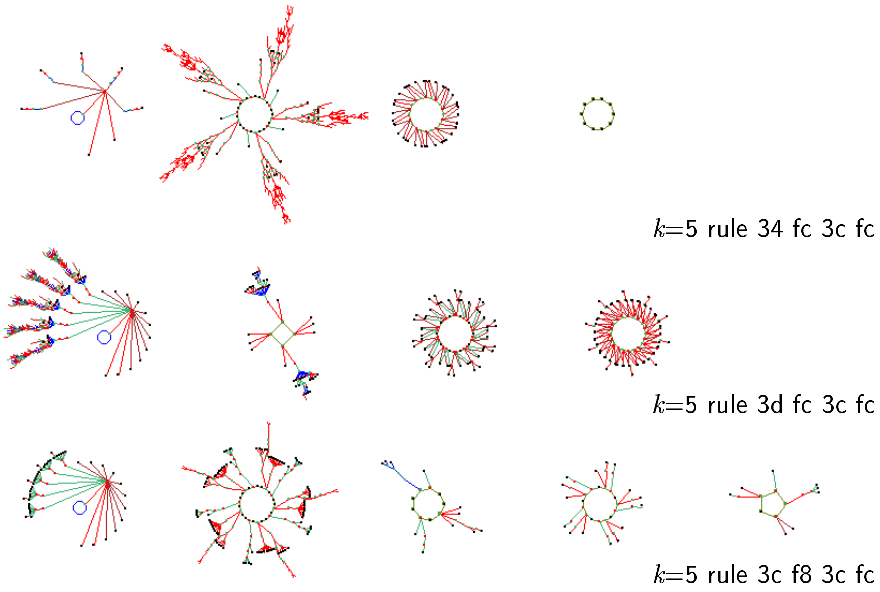


Figure 28.3: Mutation of the basin of attraction field by one bit-flip. The k=3 rule (dec) 110 was first tranfomed to the equivalent k=5, rule (hex) 3c fc 3c fc (see section 18.9). The k=5 CA, n=10 (basin field shown in figure 26.1), was then mutated in sequence by a single bit from the left (non-cumulatively). Three mutant basin fields are shown for the 5th, 8th and 14th bit-flip. The k=5 rule numbers are shown in hex.

The start and end position of the bit-flips can be altered from the default at the extreme ends of the rule-table. The following top right prompt is presented,

specify start - end position of bitflip (countdown)
start position (default 31): end position (default 0): *(this example for k=5)*

Enter the new start position, followed by the new end position. Once the end position is reached, the next mutation will revert to the start.

28.6 No pause before next mutant

When an attractor basin is complete, the program will pause and wait for **return** to be entered for the next mutant attractor basin. To suppress the pause and produce a continuous display of mutant attractor basins, the following top right prompt is presented,

dont pause after mutant graphics-y:

Enter **y** to suppress the pause. There will be a short delay to see the complete attractor basin before the next mutant is generated. The default delay cane be changed, The following subsequent prompt is presented if *y* was selected above,

change delay, enter factor (now 1.00 max 10):

Enter a factor to change the delay. For example to make it half as long enter .5, twice as long enter 2. To restore the default setting enter 1.

This option works for all types of attractor basins, basin fields, single basins, and subtrees, including a range of sizes set in section 7.1.

To interrupt the continuous display enter **q**.

Chapter 29

Final options for attractor basins

This chapter describes a number of final options before attractor basins are drawn, which depend on the parameters set previously in chapters 24 to 28.

If **s** for a "single basin/subtree" was selected at the first DDLab prompt in section 6.1 (a seed would also have been set), then a choice will need to be made between a subtree and a single basin. Its also possible to select "forwards only" at this stage (as well as in section 24.1), in which case the options in this chapter will not apply, and the program will proceed with the options for just space-time patterns as described in chapter 31.

For a subtree, its possible (and usually advisable) to run forward a given number of time-steps from the selected seed, and generate the subtree from the state reached. For a single basin the default repeat check limit to find the attractor can be amended. In both cases the number of backward steps in the subtree, or in attractor trees, can be limited to see just the core behavior of large networks.

In general, for basin of attraction fields as well as single basins and subtrees, there are further options to use other than the default reverse algorithm and to view the algorithm's inner workings (especially for RBN), to apply sequential instead of synchronous updating, to generate the neutral order components for sequential updating, and to generate (biased) random maps.

All these option have default setting and may be skipped by entering **return**.

29.1 Subtree or single basin

If a single basin or subtree was selected at the first DDLab prompt in section 6.1 (a seed would also have been set in chapter 21), a top right prompt allows either running backward to generate a subtree, or running forward to determine the attractor cycle and generate a single basin of attraction (or to select just space-time patterns).

backward for subtree-b, forward for basin (def):

Enter **return** for a single basin, **b** for a subtree. Select a subtree if a "neutral order component" is to be generated as described in section 29.8.

29.2 Subtree: Run forward before running backwards

If **b** is entered for a subtree in section 29.1 above, a further top right prompt is presented to first run the network forward from the seed state by a specified number of time-steps before running backwards. The state reached on the forward run becomes the root of the subtree. Running forwards before running backwards is usually necessary, other than for very chaotic networks such as CA with "chain" rules, see section 16.7. For most networks, a seed selected at random is very likely to be a garden-of-Eden state which has no pre-images thus no subtree.

forwards before backwards?
how many steps (def 0, max 90): *(max steps depends on n)*

Enter the number of forward steps, or accept the default, 0.

29.2.1 Backward space-time patterns

A further prompt allows changing the current setting for showing or suppressing backward space-time patterns (see section 24.7), which would normally appear on the left of the screen,

NO stp-n:
or
SHOW stp-y:

29.3 Single basin

Enter **return** in section 29.1 to select a single basin, the following top right prompt is presented,

history limit def=10,000 steps (max 65535)
draw basin-ret (or enter revised limit)
space-time patterns only (no basin)-s:

29.3.1 Select just space-time patterns, not attractor basins

Enter **s** to generate space-time patterns (running forward) only, not attractor basins. Further options in this chapter will not apply, and the final options for just space-time patterns will be presented as described in chapter 31. This is the second chance to select just space-time patterns, the first was in section section 24.1.

29.3.2 Single basin history limit

To generate a single basin the network must first be run forward from the initial state (seed) to find the attractor. States at each successive time-step are added to a history array. The state at each new time-step is checked against this history looking for a repeat which would define the attractor cycle. Its possible to change the default maximum number of time-steps held in the history array. This may be useful for some rules (such as k=3 rules (dec) 60 and 150) which are known to have very long attractor cycles with either no transients, or extremely short transients of length 1, so the history limit could be set to 2, speeding computation. Conversely, when looking for attractors with very long transients (such as k=3 rules (dec) 30 and 45, which also have very long attractor cycles), the default history limit may need to be increased.

The default maximum number of steps in the history array is 10,000. Enter the new history limit at the prompt in section 29.3 (the maximum is 65535, an unsigned short int). If zero is set, just space-time patterns will be selected as in section 29.3.1, and the forward only prompts will be presented (see chapter 31).

29.3.3 Interrupting while looking for attractor

If the dynamics is interrupted (with **q**) while the network is still iterating forward looking for the attractor cycle, the following top right message is presented,

still looking for attractor cycle, 1247 iterations
exit-q, cont-ret: *(values shown are examples)*

Try this with the k=3 rule (dec) 30 which has very long period attractors cycles. Note that if the history limit is set shorter than the transient length the attractor will never be found.

29.3.4 Run-in greater than history

An unusual situation may arise if the history limit is set smaller than the eventual "run-in" length, the number of time-steps in the transient plus the attractor period. A repeat (and the attractor) may be found, but there may be insufficient states in the history array to define the attractor. In this case the program will stop with the following message,

att period=91, run-in=111 *(values shown are examples)*
history limit=44 (too small), should be ≥111, cont-ret:

Enter **return**. The message in section 29.3.3 will appear. Enter **q** to exit the attractor basin drawing routine. Try this with the k=3 rule (dec) 110, n=14, with a seed where one bit is set to 1, the rest 0, and the history limit set smaller than 111.

29.4 Final options before drawing attractor basins

Before drawing attractor basins, a final top right prompt is presented as a reminder of the type of network and attractor basin, and giving further special options. Enter **return** to skip these options and generate the attractor basin.

The exact wording of the prompt depends on the network parameters, for example,

for a CA basin of attraction field
basin field (local) n=10, exh-e rmap-r seq-s nto-o
view ppstack-1:

for a RBN sub-tree
sub-tree (non-local) n=8, exh-e rmap-r seq-s nto-o
view ppstack-1 +pause-2, reorder(on) tog +R, limit backsteps (add)-b:

The first line of the prompt includes a reminder of the following,

subtree, **single basin**, **basin field** ...	the type of attractor basin
n=10, **n=5-12** ...	the network size n, or the start and finish sizes for a range of sizes
(local), **(non-local)**	the wiring; whether local (as in 1d CA) or non-local (as in RBN, but including regular 2d and 3d.)

Other options are as follows, some depending on the network and type of basin,

exh-e ...	to select the exhaustive testing algorithm, for $n \leq 31$ (see section 29.5).
rmap-r ...	to select a random map, a random graph with out-degree 1, uses the exhaustive testing algorithm, for $n \leq 31$ (see section 29.6).
seq-s ...	to select sequential updating, uses the exhaustive testing algorithm, for $n \leq 31$ (see section 29.7).
nto-o ...	to find neutral orders in sequential updating, for $n \leq 12$, for a subtree or basin field only, not for a single basin of attraction (see section 29.8).
view ppstack-1 ...	to view the changing partial pre-image stack, indicating the inner workings of the CA or RBN reverse algorithm (see section 29.9).
+pause-2 ...	for RBN only, to view the partial pre-image histogram, as above, and also pause it after each set of valid pre-images (see section 29.9.2).
reorder(on) +R ...	for RBN only, the reverse algorithm reorders the cells for more efficient computation. The reordering can be toggled off/on by entering **R**, or another entry followed by **R**, for example **1R** (see section 29.10).
limit backsteps +b ...	to restrict the number of backward steps, for a single basin or sub-tree only, enter **b** or another entry followed by **b** (see section 29.11).

These various special options are described below. They can be skipped by pressing **return**.

29.5 Exhaustive testing

Enter **e** in section 29.4 to use the exhaustive testing algorithm for generating pre-images instead of the default direct algorithms (see section 2.12). There are two direct reverse algorithms[12, 18]. The 1d CA algorithm is specifically for networks with 1d local wiring (i.e. 1d CA, but which may have mixed rules). The RBN algorithm is a general algorithm for networks which may have heterogeneous rules, wiring and k, but which also works for CA, a subset of RBN. The RBN algorithm is also used for 2d and 3d CA.

The exhaustive algorithm is even more general. It works for random graphs with out-degree one (or random maps, see section 29.6), including RBN and CA, which are subsets of random maps.

The exhaustive testing algorithm is limited to small networks. It is efficient for basin of attraction fields, or large basins that use up a good proportion of state-space, and additionally is not sensitive to neighborhood size, k. However the method is highly sensitive to increasing network size n. Every state in state-space must be iterated forward by one step, and the list of successors held in an array. Effectively a list of 2^n pairs of states is created, each state and its successor. Pre-images are found by scanning this list. The list can be saved and loaded from a `.exh` file. The absolute maximum n=31, but in practice n should be limited to smaller networks because of memory and time constraints.

If this option is selected, the following prompt near the top right is presented,

exhaustive pairs: load-l, compute-ret, and save-s:

29.5.1 Loading the exhaustive pairs

Enter **l** in section 29.5 to load a previously saved list of exhaustive pairs from a `.exh` file (see Filing, chapter 35), and draw the attractor basins. Note that the current value of n must equal the file value of n. If not a top right notice such as the following will be displayed,

file n=7 != network n=8 (wrong file) cont-ret: *(values shown are examples)*

29.5.2 Generating the exhaustive list

Enter **return** in section 29.5 to compute the exhaustive list. A horizontal red progress bar near the top right of the screen will monitor the proportion of state-space completed so far, together with the following message,

setting up exhaustive pairs, interrupt-q::

If **q** is entered to interrupt, which may be necessary if you loose patience with a large network, the following prompt is presented,

exhaustive pairs 7% complete; stop-q options-o cont-ret:

Enter **q** to abandon the exhaustive pairs computation, enter **o** for the "options" prompts in section 30.4, or **return** to continue.

29.5.3 Saving the exhaustive pairs

Enter **s** in section 29.5 to compute and then save the exhaustive list. Once the list is generated the following prompt is presented below the red progress bar,

SAVE exhaustive pairs-s:

Enter **s** to save the list as a `.exh` file (see Filing, chapter 35).

29.6 Random map

A random map is a directed graph with out-degree one, representing an an exhaustive list of transitions, and is arguably the most general discrete dynamical system. The set of all random maps contains the subset of all RBN, which in turn contains the subset of all CA.

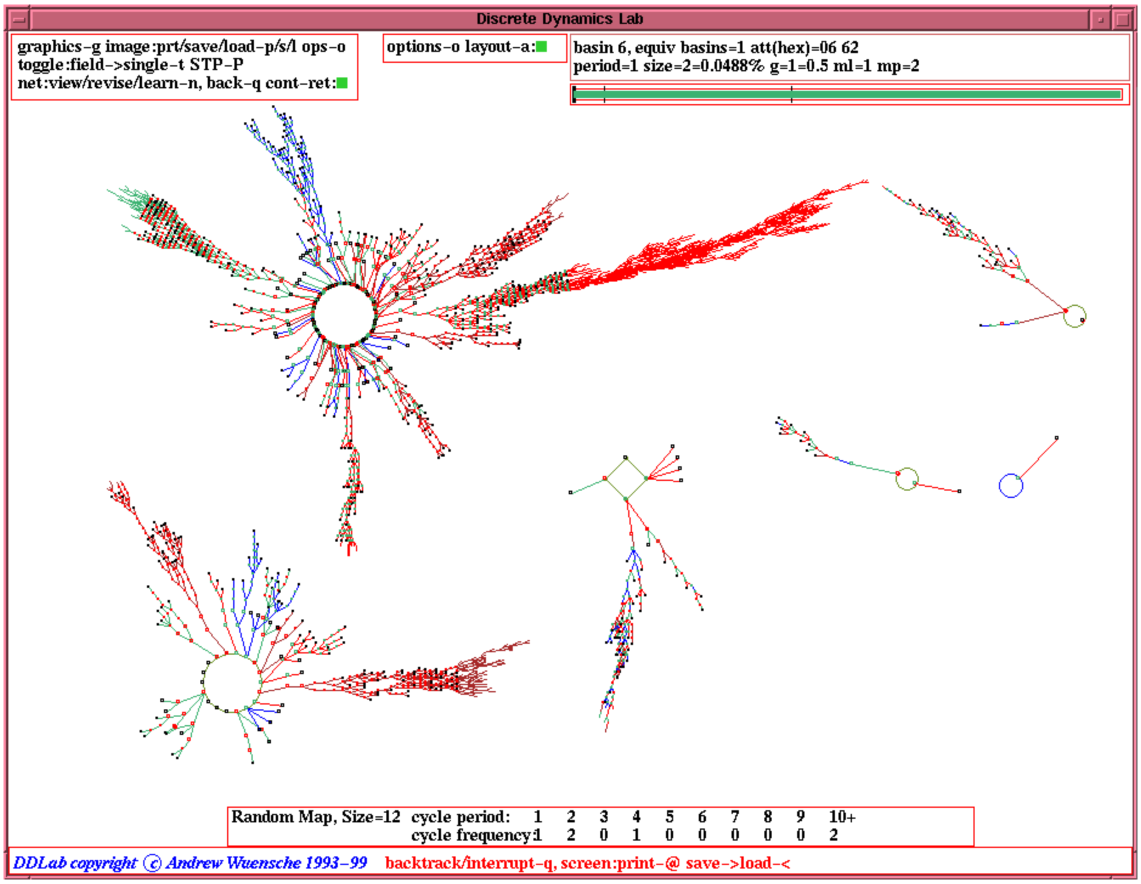


Figure 29.1: The basin of attraction field of a random map (with no bias), $n = 12$. A characteristic giant component is evident. On-the-fly layout options in section 25.3.2 and 25.3.3 were used in this example to reduce overlap between basins.

Provided that the size of the map is a power of 2, the function to set up a random map is implicit in the exhaustive testing algorithm described in section 29.5. This algorithm creates a list of 2^n pairs of states, each state and its successor, by iterating each state in state-space forward by one step according to the network dynamics. A random map, on the other hand, assigns the successors at random (or with some bias). Assigning successors according to the dynamics of a particular class of network is just a special case of such a random assignment.

Some randomly assigned successors states are likely to occur more than once, so other states will not occur at all (there will be no room left in the list). These are the states without pre-images, "garden-of-Eden" states. A random map has all the properties of a dynamical system, trajectories, attractors, etc. though this may be irreduceible to a sparsely connected network. A fully connected network, however, where $k = n$, where each cell receives inputs from all cells (including from itself), can implement any random map. The space of all such fully connected networks, and all random maps (of the Boolean hypercube of length n) is $(2^n)^{2^n}$, which is also the upper limit for the space of all possible basin of attraction fields.

Enter **r** in section 29.4 to generate a random map. The option is only active if $n \leq 31$, but in practice n should be smaller because of memory and time constraints. The following prompt near the top right is presented, similar to the prompt in "Exhaustive testing", section 29.5,

random map: load-l, compute-ret, and save-s:

29.6.1 Loading the random map

Enter **l** in section 29.6 above to load the exhaustive list of pairs defining the random map from a previously saved `.exh` file, described in section 29.5.1.

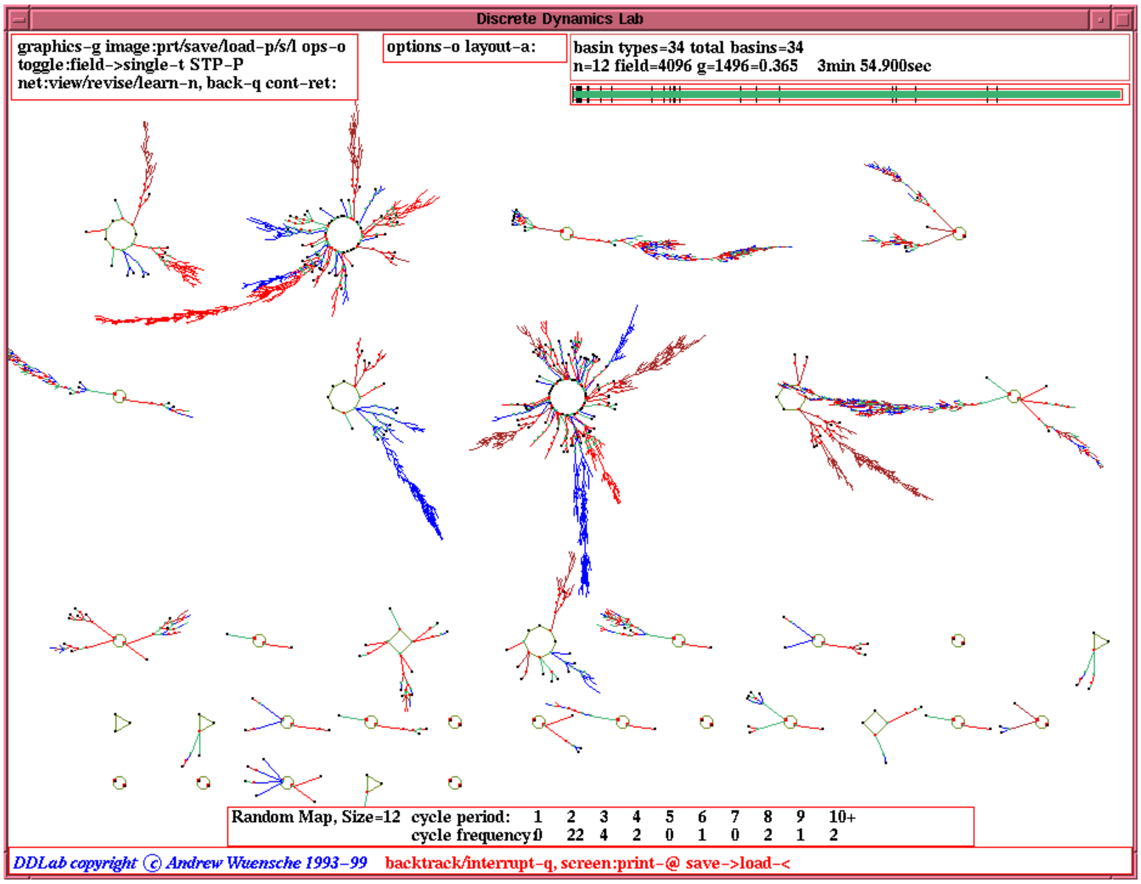


Figure 29.2: The basin of attraction field of a random map with Hamming bias 2, $n = 12$. Compared with a random map with no bias, there are typically more basins with shorter periods. On-the-fly layout options in section 25.3.2 and 25.3.3 were used in this example to reduce overlap between basins.

29.6.2 Generating the random map

Enter **return** in section 29.6 to assign the map at random, or to bias the random map according to Hamming distance. The following prompt near the top right is presented,

random map: rand-def, or set Hamming distance (0-8): *(for n=8)*

Enter **return** for a random map, or enter the Hamming distance (0-8), so that exhaustive pairs differ by that number of bits, i.e. between each bitstring and its assigned successor. Figure 29.1 shows a random map with no bias, with the expected 'giant component" from graph theory. Figure 29.2 shows a a random map biased so that successors differ by just 2 bits from their pre-images (Hamming distance 2).

While the exhaustive pairs are being assigned, a horizontal red progress bar near the top right of the screen will monitor the proportion of state-space completed so far, together with prompts to interrupt etc. as described in section 29.5.2.

29.6.3 Saving the random map

Enter **s** in section 29.6 to both assign and save the random map. The exhaustive list will be created, and may be saved as described in section 29.5.3.

29.6.4 Random map data

I addition to the normal data shown in a top right window when a subtree, basin, or basin of attraction fields is complete (see chapter 27), for a random map, additional field data shows the frequency of different cycle periods in a bottom center window, which is of some interest in graph theory[16]. This example (from figure 29.2) shows that there are 22 cycle with period 2, 4 with period 3, etc., and 2 with period 10 or more,

Random Map, Size=12 cycle period: 12345678 910+
cycle frequency: 022420102 12

29.7 Sequential updating

By default, network updating is synchronous, in parallel. Alternatively, the updating can be sequential. There are $n!$ possible sequential update orders in a network of size n. DDLab provides simple default orders, forward or backward along the network index, or a random order. A specific order can also be set by hand. The order can be saved and loaded from a `.ord` file. In addition, all possible orders can be listed, with an order index from 0 to $n!$-1. An order can be chosen from this list.

Enter **s** in section 29.4 to select the order. The following top right prompt is presented,

sequential updating:->(def) <-b rand-r set-s all-S:

Once the sequential order has been is selected, the attractor basins will be drawn. A state in a sequential network is the pattern of 0s and 1s when the n updates are complete, intermediate updates are not included. Finding pre-images for sequential updating uses the exhaustive testing algorithm (see section 29.5), so the maximum size of networks is 31, in practice smaller.

29.7.1 Left to right or right to left orders

Enter **return** in section 29.7 above for the default sequential updating order, from left to right in a 1d network, or in general counting down from the cell index $n-1$ to 0 (see section 9.6 for the indexing conventions). Enter **b** for the opposite (backward) order, from right to left and the cell index 0 to $n-1$.

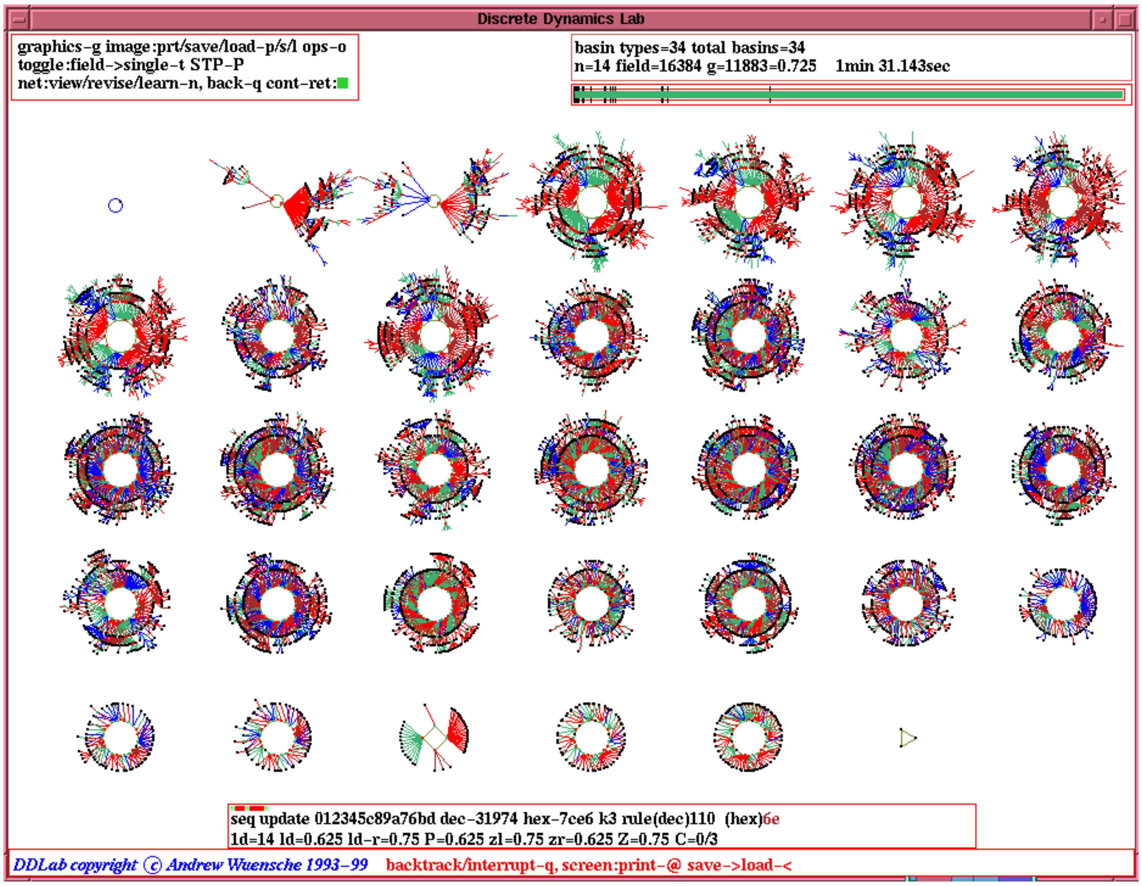


Figure 29.3: A basin of attraction field for a 1d CA with sequential updating, n=14, k = 3 rule (dec) 110. The sequential order is indicated in the lower rule window

29.7.2 Random order

Enter **r** in section 29.7 to set a random order. A top right prompt showing the random order, similar to the following, is presented,

randomly re-ordered sequence, startindex=13
5 4 6 2 12 10 11 8 1 13 0 3 7 9 *(for n=14, values shown are examples)*
seq shown from cell index 13-0
save/load-s/l re-order-r cont-ret:

The *position* of the numbers represents the cell index, the numbers themselves (0 to $n-1$) define the update order of the cell at that position. In the example above, the cell at index 13 (far left) is ordered 5, the cell at index 0 (far right) is ordered 9. Enter **s** to save the order to a `.ord` file **l** to load an order (see Filing, chapter 35). Enter **r** to select a new random order, and **return** to accept the order.

29.7.3 Set specific order

Enter **s** in section 29.7 to set a specific order. The order for each cell (from left to right) is set with the following top right prompt,

> **enter sequence (no repeats) 0-13 or random (def)-r back-b**
> **index 9: 4 8 10 3 * * * * * * * * * *** *(for n=14, cells indexed 13-0. for example)*
> *(cells at index 13,12,11,10 have been set, now prompting for index 9, values shown are examples)*

A prompt for each cell index from $n-1$ to 0 is presented in turn. ***'s indicate unallocated cell indices, numbers show allocated cell indices. Enter a unique order for each cell index (repeats will be rejected). **return** gives a valid random order. Enter **b** to backtrack by one cell. When the selected order is complete, a unique number (the order index) is shown in decimal and hex. The order index specifies and codes for the order itself. Further options are also presented, for example,

> **enter sequence (no repeats) 0-13 or random (def)-r back-b**
> **index 0: 4 12 10 3 7 6 11 9 1 2 0 5 13 12 =27841(dec)=6cc1(hex)**
> **save/load-s/l re-order-r cont-ret:** *(values shown are examples)*

Enter **s** to save the order to `to a .ord` file, **l** to load an order (see Filing, chapter 35). Enter **r** to reselect, **return** to accept the order.

29.7.4 List all orders

Enter **S** in section 29.7 to list all $n!$ possible orders. As many orders as will fit will be displayed in a window on the right of the screen, followed by further options. For example, for n=14, the first two orders and last two orders in the list are shown below,

> **all possible orders n=14, 0-1278945279**
> **dec index - order - hexindex**
> **0 - 0123456789abcd - 0** *(this is the left to right order)*
> **1 - 0123456789abdc - 1**
> ... *(the list continues, the last two order are shown below)*
> **1278945278 - 03a17cdb48529- 27fe**
> **1278945279 - 03a17cdb48592- 27ff** *(this is the right to left order)*
> **quit-q jump-j copy-c set-s accept-a more-ret:**

The center column is the order (in hex), where the the *position* of the numbers represents the cell index, the numbers themselves (0 to $n-1$) define the update order of the cell at that position. The left and right columns show the order index (0 to $n!$) which specifies and codes for the order. The left column is in decimal, the right in hex.

Enter **return** to show more entries in a long list in successive windows. Enter **q** to exit the list (**q** again to backtrack). Entering **return** after **q** will generate an attractor basin with *synchronous* updating, but using the exhaustive algorithm. Various further options are explained in sections 29.7.4.1 29.7.4.4 below.

29.7.4.1 List all orders - jump

Enter **j** in section 29.7.4 above to jump to a new initial order index, the following prompt appears at the bottom of the list,

> **jump to index (0-1278945279):** *(i.e. for* $n = 14$, $0 - 14! - 1$*)*

29.7.4.2 List all orders - copy

Enter **c** in section 29.7.4 to "copy" an order from the list, the following prompt appears at the bottom of the list,

> **select by index (0-1278945279):** *(i.e. for* $n = 14$, $0 - 14! - 1$*)*

Enter a decimal order index, the list will be redrawn with that index at the top. Then enter **a** to accept the selection, and draw the attractor basin.

29.7.4.3 List all orders - set order seed

Enter **s** to set the order index as if it were a seed in section 21. The "order seed" prompt will appear (similar to the prompt in section 21.1) as follows,

> **Select order seed, max-1278945278 win-w empty-e fill-f dec-d rand-r bits1d-b bits2d-B hex-h (def-r):**

Select the order seed as if it were a network seed in the various ways described in chapter 21. Note that some functions in this section may be irrelevant in the context of the order seed. Enter **a** to accept the selection, and draw the attractor basin.

29.7.4.4 List all orders - accept

Then enter **a** to accept the selection in sections 29.7.4.2 or 29.7.4.3, and draw the attractor basin. If a selection has not been made, entering **a** will select the order index 0, i.e. sequential updating from left to right, from cell index $n - 1$ to 0.

29.7.5 Drawing the attractor basin with sequential updating

Once the sequential order has been set, the list of exhaustive pairs will be generated as described in section 29.5, and the attractor basin will be drawn. At the same time, the "rule window" will be displayed at the bottom of the screen, giving the CA rule (or the rule at index 0 for an RBN), and other information described section 16.14, but in addition the updating order and the order index (in decimal and hex), for example,

> ·■■·■■■· *(the rule look-up table shown as a bit pattern, values shown are examples)*
> **seq update 012345c89a76bd dec-31974 hex-7ce6** k**3 rule-(dec)110 (hex)6e**
> **1d=14 ld=0.625 ld-r=0.75 P=0.625 zl=0.75 zr=0.625 Z=0.75 C=0/3**

29.8 Neutral order components

for $n \leq 12$

In sequential updating, different orders may produce equivalent dynamics. Its of some interest from a mathematical perspective to compute the neutral order components, how sequence space is divided up into sets of orders with identical dynamics. This turns out to depend on the network's wiring scheme. The algorithm in DDLab that computes neutral order components (limited to network size $n \leq 12$), does so by starting with a particular order (the order seed), and generating its equivalent orders by swapping the order of pairs of neighboring cells that have no links to each other, neutral pairs. Valid swaps within the first order are treated as "pre-images", and these pre-images may have pre-images in turn (repeat orders are discounted). When no valid swaps remain the neutral component is complete.

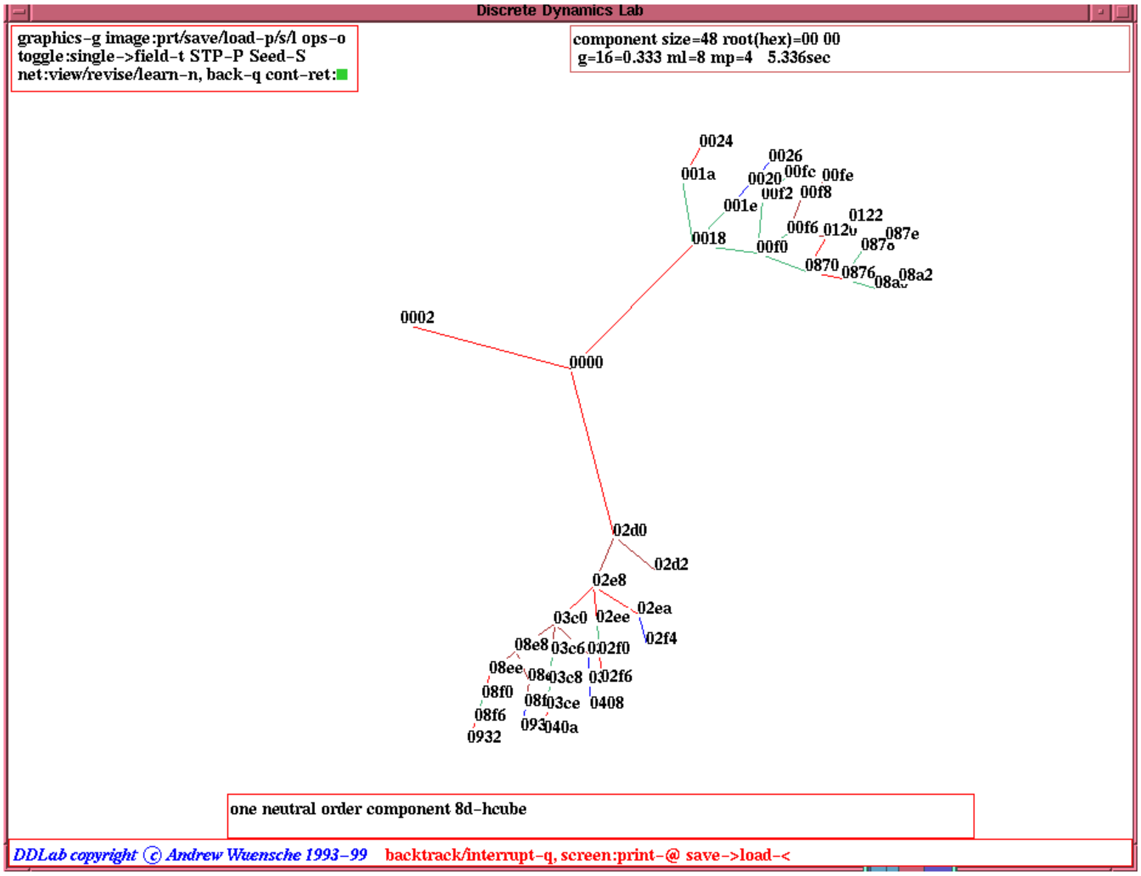


Figure 29.4: A neutral subtree, a set of equivalent orders generated from the order root, computed and drawn in an analogous way to a subtree for network dynamics. Nodes (shown in hex) represent the order index, a unique number for each order. This example is for n=8, k=3, with 8d-hypercube wiring (see section 11.2). The order seed=0, i.e. the order is 76543210. The subtree is the first neutral component in the neutral field in figure 29.5.

DDlab treats these components as subtrees. A single component subtree, or the entire set of components subtrees making up sequence space (the neutral field) can be computed and drawn in an analogous way to a single subtree or basin of attraction field for network dynamics. The layout and node label options described in chapter 26 apply.

Nodes are labeled according to the order index (a unique number, 0 to $n!-1$ (see section 29.7.4), in decimal, hex, as a bitstring, etc as previously selected in section 26.2. The default is as a spot.

Enter **o** at the prompt in section 29.4 to generate the neutral subtree or neutral field, depending on whether a subtree or basin of attraction field is currently selected (the prompt **nto-o** is not activated if a single basin of attraction was selected in section 29.1).

Note that the rule or rule scheme, or a subtree's initial state, selected previously, plays no role in computing neutral order components.

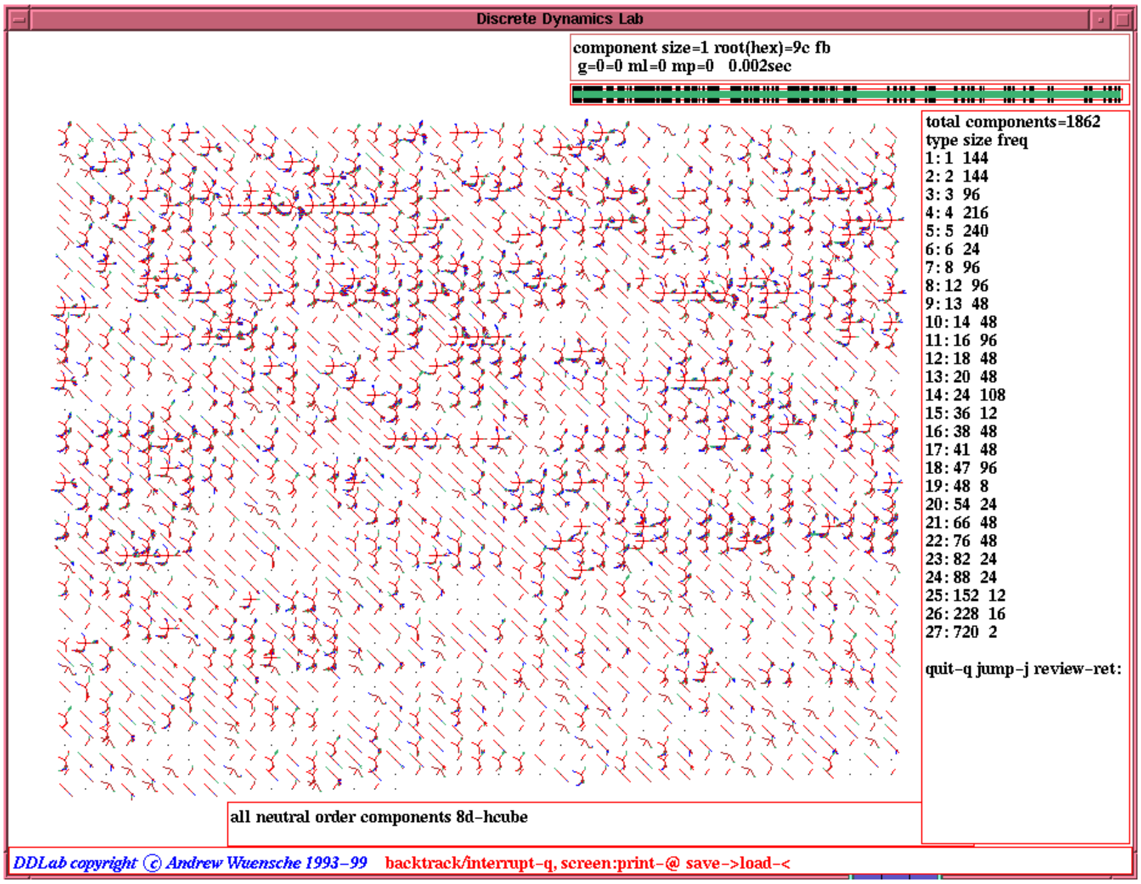


Figure 29.5: The neutral field, the entire set of components subtrees making up sequence space, computed and drawn in an analogous way to a basin of attraction fields for network dynamics. The total number of separate components (drawn as "subtrees"), and a list of component sizes and the frequency of different sizes is shown in a window on the right once the neutral field is complete. This example is for n=8, k=3, with 8d-hypercube wiring (see section 11.2). The first subtree (top right, order seed=0) is shown in detail in figure 29.4

29.8.1 Neutral subtree

If **o** is entered in section 29.4 for a neutral subtree, a lower center window prompts for the neutral order seed, or subtree root,

> **Select order seed, max=40318, win-w empty-e fill-f rand-r dec-d bits1d-b birs2d-B hex-h (def-d):** *(max=$n!-1$, in this example n=8)*

Enter an order index, 0 to $n!-1$ (see section 29.7.4) as the seed, in decimal, hex, as a bitstring, etc. The selection works in the same way as selecting a network seed, described at length in chapter 21. A selection greater than $n!-1$ will be rejected, with the message,

> **order index 44444 > max (8!=4318) cont-ret:** *(for example)*

The following examples relate to figure 29.4, for n=8, k=3, with 8d-hypercube wiring, (see section 11.2) As the subtree is dawn, a message appears in a window at the bottom of the screen,

> **one neutral component 8d-hypercube**

When a subtree is complete, a top right window gives some data on the subtree as follows,

> **component size=48 root(hex)=00 00** *(values shown are examples)*
> **g=16=0.333 ml=8 mp=4 5.336sec**

29.8.2 Neutral field

If **o** is entered in section 29.4 for a neutral field, all neutral order components making up sequence space will be generated, as in the example in figure 29.5. The order components are drawn as "subtrees" (see also figure 29.4), and the data on each subtree appears in a top right window while the next subtree is drawn (section 29.8.1 describes the data).

Its a good idea to drastically reduce the scale of the subtrees in section 25.2 as there are usually many small neutral order components.

Once the neutral field is complete, component types are listed according to their size, and the frequency of different sizes. A list of as many component types as will fit will be displayed in a window on the right of the screen, followed by further options. For example, for n=8, k=3, with 8d-hypercube wiring, shown in figure 29.5,

total components=1862
type size freq
1: 1 144
2: 2 144
3: 3 96
4: 4 216
... *(the list continues)*
25: 152 12
26: 228 16
17: 720 2
quit-q jump-j review-ret: *(if the complete list fits the window, as in this case)*
or
quit-q jump-j more-ret: *(if more than one window is required)*

Enter **q** to quit the list, **j** to jump to a new type (as in section 29.7.4.1, and **return** to re-list from the start, or to see more of a long list.

A vacant space may be left for the list window so as not to hide some of the graphics output, by setting the minimum right border at a high value in section 25.2.8.

29.9 Viewing the partial pre-image stack

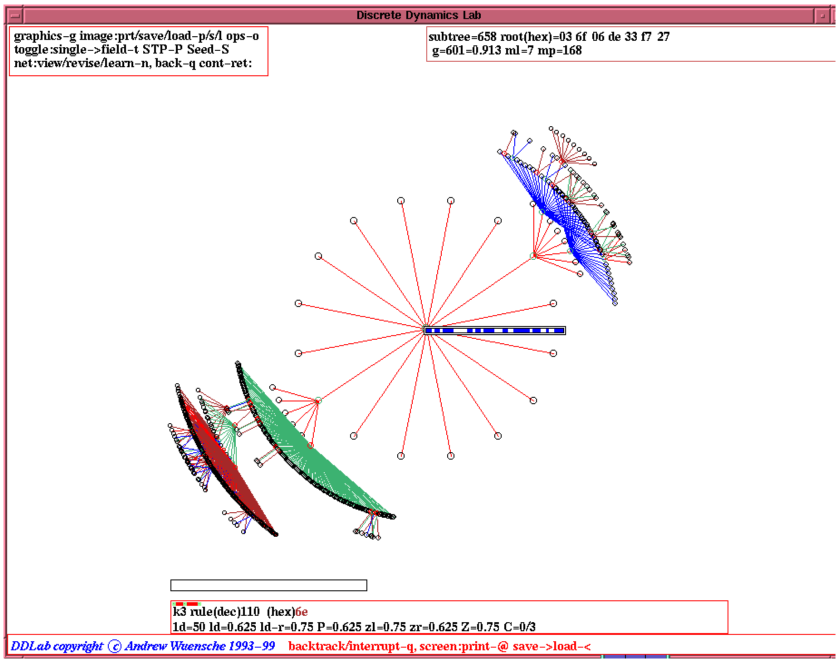


Figure 29.6: A CA subtree, showing the partial pre-image stack. $n = 50$, $k3$ rule (dec) 110. The partial pre-image stack is represented by a horizontal bar within the rectangle at the bottom of the screen. The bar changes as the partial pre-image stack grows and shrinks. Its length (in pixels) indicates the varying size of the partial pre-image stack.

For synchronous networks, different reverse algorithms to generate pre-images[12, 13, 18] apply to local and non local wiring. The local algorithm is for networks with 1d CA (nearest neighbor) wiring with homogeneous k, but with possibly mixed rules. The non-local algorithm is for any other type of wiring (RBN, 2d or 3d networks). The non-local algorithm in general is considerably

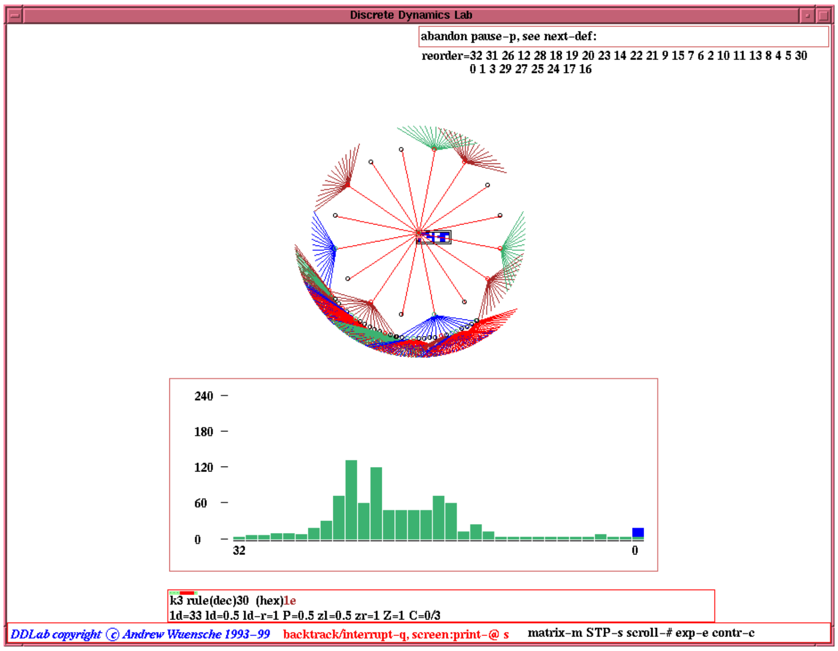


Figure 29.7: A RBN subtree, showing a snapshot of the changing partial pre-image histogram in a window at the bottom of the screen, for a RBN, $n = 33$, homogeneious k=3 rule 30. The final bar height represents the number of pre-images of a particular state. The numbers below the top right prompt window show the reorderd cells to optimize the RBN reverse algorithm (see section 29.10.

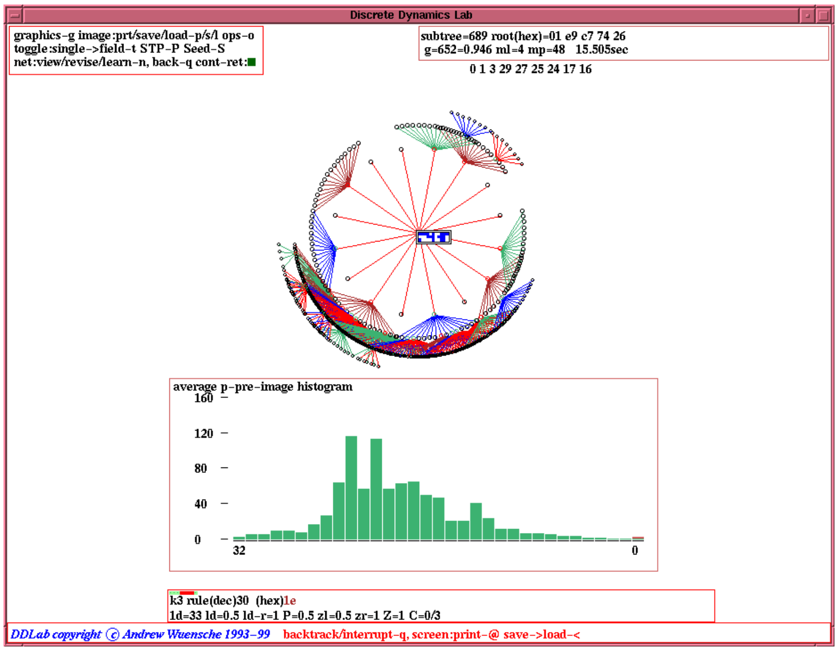


Figure 29.8: When the subtree is complete (or interrupted), a final histogram gives the average over all histograms to date. For the same network as in figure 29.8.2 above.

slower. Both algorithms work by generating a set of partly computed pre-images, "partial pre-images", held in a stack until either completed or rejected as invalid. Each partial pre-image can give rise to more partial pre-images.

Some inner workings of these algorithm may be observed by viewing the partial pre-image stack as it initially grows, then shrinks, both for local wiring, and for non-local wiring where the the partial pre-image stack is presented as a histogram. These options are described below.

29.9.1 Partial pre-image stack, local wiring

Enter **1** in section 29.4, to see how the partial pre-image stack varies in size using the local wiring reverse algorithm. This is shown as a horizontal bar near the bottom of the screen. The length of the bar (in pixels) indicates the varying size of the partial pre-image stack.

29.9.2 Partial pre-image stack, non-local wiring

Enter **1** or **2** in section 29.4 for an inner view of the workings of the non-local reverse algorithm. A histogram will appear in the lower half of the screen with n columns, showing the size of the partial pre-image stack for each successive cell in the network starting from the left. When the attractor basin is complete (or abandoned), a final histogram gives the average over all histograms to date.

If **1** is entered in section 29.4, the histogram changes continually, and the following top right prompt is predented,

partial pre-image histogram: start pause-p:

If **2** is entered in section 29.4, or if **p** is entered above, there will be a pause after each pre-image fan (but not for garden-of Eden states), and the following top right prompt is presented,

abandon pause-p, see next-def:

Generally, successive values in the histogram tend to increase up to a middle zone, then decrease. If the value reduces to zero before the last column is reached, the state is a garden-of-Eden state (the histogram will not pause). The value of the last green column represents the final set of pre-images. However these pre-images may contain "wild-card" cells due to cells in the network that have no outwires. An additional red column on top of the last green column indicates additional pre-images that result from filling in these wild-cards.

29.10 Reorder to optimize the RBN reverse algorithm

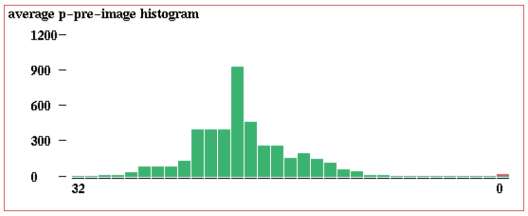


Figure 29.9: The final partial pre-image histogram, with the reordering algorithm suppressed, for the same RBN as in figure 29.8.2, $n = 33$, homogeneious k=3 rule 30. An approximate comparison of the time t, and the maximum average stack height h, with the reordering algorithm active is as follows, active: t=15sec, h=120, inactive: t=1min 14sec, h=1200.

The RBN reverse algorithm[13, 16] builds a stack of partial (i.e. incomplete) pre-images by taking each cell in turn and filling in valid entries taken from the rule-table of that cell according to its wiring. Any conflict between successive entries disqualifies the partial pre-image. The partial pre-image stack will grow initialy, then shrink as conflicts mount, (see section 29.9.2).

The algorithm works for cells taken in any order, for example left to right. However, to reduce the initial growth of the partial pre-image stack, the cell order is reordered to ensure there is wiring overlap of succesive cells (if possible), to allow conflics to happen early. This reduces the growth of the partial pre-image stack considerably compared to a simple left to right order, thus reducing memory load and speeding computation. The new order is shown below the top right prompt window, as in figure 29.8.2 for n=33, where the order from 32-0 has been reordered as follows,

32 31 26 12 28 18 19 20 23 14 22 21 9 15 7 6 2 10 11 13 8 4 5 30
0 1 3 29 27 25 24 17 16

This reordering has reduced the time (and memory) to complete the subtree by a factor of 10 (see figure 29.9.2. If there is already substantial overlap in the wiring relative to a left to right order, the improvement will be less.

The new order is local within the reverse algorithm and makes make very little difference to the presentation of attractor basins. The states remain the same. The precice arrangement of states in the attractor basin may be slightly different. The partial pre-image histogram in section 29.9.2 will appear according to the new order.

The algorithm to reorder is implimented by default, however if can be toggled off/on by entering **R**, or another entry followed by **R**, for example **1R** in section 29.4. Toggling reordering off impliments the reverse algorithm from left to right, as in versions of DDLab prior to August 2000.

29.11 Limit backward steps

Its possible to limit the number of backward steps in subtrees (from the root state), or in the trees of single basins (from the attractor), to see just the core behavior, which may be useful for large networks.

If **b**, or another entry followed by **b**, is entered in section 29.4, the following top right prompt is presented,

enter max number of backward steps (def no limit):

Enter the number of backward steps. Trees and sub-tree will stop generating further pre-images when the number is reached. Enter **return** not to impose a limit.

Chapter 30

Drawing attractor basins, and changes on-the-fly

Following the final options described in section 29.4, the drawing of attractor basins will commence, according to the current presentation, layout and other settings. While they are being drawn, some of these settings can be reset, either by pausing the drawing and responding to further prompts, or immediately, on-the-fly, by various key hits. Many network parameters may also be reset without the need to backtrack to the early prompts. This chapter describes these options.

Note also that if one of various "pause" options was selected in section 27.2, the angular orientation, spacing and/or position of each successive basin, tree or subtree can be amended on-the-fly as basins are drawn. This is described in section 25.3.

30.1 The progress bar for basin of attraction fields

For basin of attraction fields only, a horizontal green progress bar below the data window (see section 27.3) indicates how much of state-space has been used up so far. A bar that over- or undershoots indicates an error. The vertical bars on the horizontal bar indicate the position of the seeds (according to their decimal equivalent values) for successive basins, starting with 0 on the left.

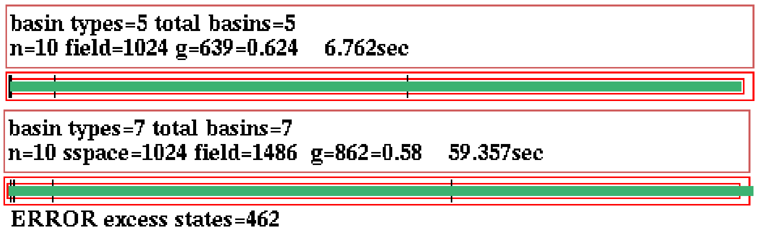


Figure 30.1: Examples of the basin of attraction field progress bar and the data window, for an RBN. *top:* For a normal run without error. *bottom:* An example where the wiring was changed during a pause, leading to inconsistent data where the size of the field is greater than state-space. The progress bar has gone off its scale on the right.

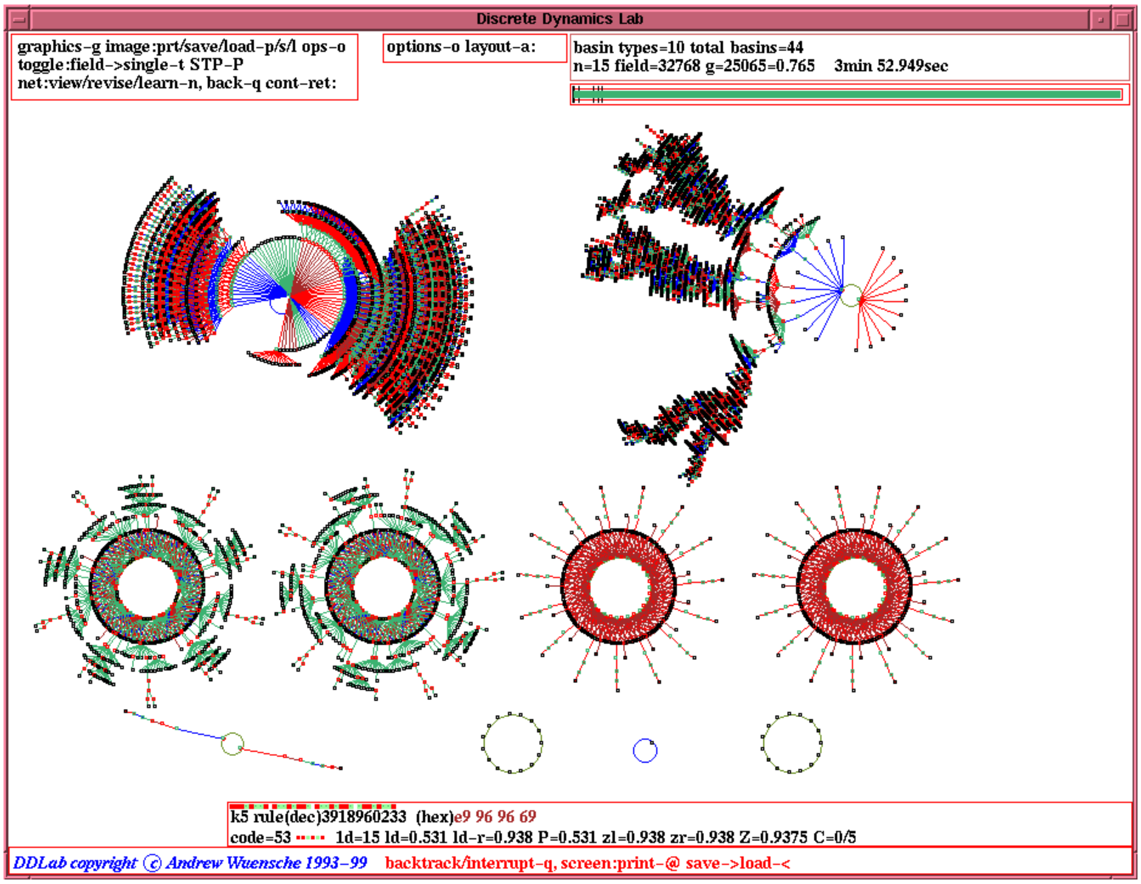


Figure 30.2: The DDLab screen showing a basin of attraction field. This example is for a 1d CA, n=15, k=5 totalistic code 53. To achieve this layout, a pause was selected after each basin in section 27.2, and the position and spacing of basins were amended on the fly as described in section 25.3. Various typical indications on the screen are as follows,
top left: The "Attractor basin complete" options window (section 30.4).
top centre: The window for amending the layout on-the-fly, (secton 25.3), if this option was set.
top right: The data window described in section 27.3.
below top right: The progress bar described in section 30.1.
bottom centre: The rule window, described in section 16.14.
foot of screen: The title bar, described in section 5.4. While attractor basins are being drawn, this also shows reminders about on-the-fly options: **matrix-m STP-s scroll-# exp-e contr-c** (see section 30.3).

30.2 Attractor basins, interrupting and changing

Enter **q** to interrupt attractor basins. If the interrupt occurred between successive pre-images within one pre-image fan (as opposed to between successive pre-image fans) the following top right message is displayed,

early exit - in pre-image fan

Then one of the following prompts is presented, depending on whether the attractor basin is a subtree, single basin or a basin of attraction field,

for a subtree
options-o, stop subtree-q, cont-ret:
for a single basin
next tree-n, options-o, stop basin-q, cont-ret:
for a basin of attraction field
next tree/basin-n, options-o, stop field-q, cont-ret:

If the in-degree histogram was set in section 24.3, a further option is offered to view the histogram as computed so far, for example,

inhist-h, options-o, stop subtree-q, cont-ret: *(for a subtree)*

Enter **h** to view the in-degree histogram (see section 24.3).

Options **n**, **o** and **q** are described sections 30.2.1 to 30.4 below. After these various options are complete, the program generally returns to the prompt in this section. Enter **return** to continue the attractor basin from the point of interrupt.

30.2.1 Abandoning a tree and continuing with the next tree

Enter **n** in section 30.2, to abandon the tree that is currently being generated and start the next tree. This also applies to a subtree from the uniform states, all 0s or all 1s (see section 26.1). If compression was not suppressed in section 26.1.1 all the equivalent trees (or subtrees from the uniform states) will be abandoned, and the next set of equivalent trees started.

Abandoning a subtree before it is complete will result in errors in attractor basin data (section 27.3). For a basin of attraction field, the progress bar is likely to go off its scale, with a message such as **ERROR excess states=19124** below it. This indicates that more states were computed than the size of state-space.

30.2.2 Other options

If **o** is entered in section 30.2, the "Attractor basin complete" options are presented in a top left window, described in section 30.4. When the implimentation of these options is finished, the attractor basin will either continue from the point at which it was interrupted, or if certain parameters were changed, including a new rule or seed, a new attractor basin will be started.

30.2.3 Errors in attractor basins

Its important to note that any changes to the network parameters made during a pause or interrupt will inevitably result in errors/inconsistencies in data and attractor basin structure. The **view/revise/learn-n** option allows the network to be changed. If changes are made during a pause, and the attractor basin is then continued, the result will be inconsistent with any given network. For a basin of attraction field, the progress bar is likely to go off its scale. The size of state-space as well as the field will be indicated in the data window. For example,

... sspace=1024 field=1486

A message will also appear below the progress bar. For example,

ERROR excess states=462

This indicates that more states were computed than the size of state-space.

Note also that although the wiring in a 1d CA may be examined and changed during the pause, the change is only temporary, and will be automatically restored to allow the attractor basin of the 1d CA to continue using the 1d CA reverse algorithm. For other networks, RBN and 2d or 3d CA this does not apply, and the wiring can be changed during a pause, but of course this will lead to inconsistencies and errors as described earlier. Rule changes during a pause will remain in effect for any network, again leading to inconsistencies and errors.

30.2.4 Abandon the attractor basin

Enter **q** in section 30.2 to abandon the attractor basin that was interrupted. The data on the incomplete attractor basin is shown in a top right window (see section 27.3), preceded by the words "EARLY EXIT" to indicate that the data applies to an incomplete attractor basin. For example,

EARLY EXIT basin types=7 totalbasins=76
n=15 sspace=32768 field=23222 g=16711=0.72

At the same time the "Attractor basin complete" options are presented in a top left window, described in section 30.4.

30.3 Attractor basin options, on-the-fly

A number of key hits allow changing the presentation of attractor basins on-the-fly. A reminder of some of these are shown in the bottom title window.

matrix-m STP-s scroll-# exp-e contr-c ppstack-p
*(***ppstack-p** *if the partial pre-image histogram was set in section 29.9.2)*

The commands only activate between pre-image fans, so the key may need to be hit more than once. The following key hits will make changes as follows,

on-the-fly key hit	*what it means*
matrix-m ...	Toggle the state-space matrix on and off, see section 24.2.
STP-s ...	Toggle backwards space-time patterns on and off, see section 24.7.
exp-e contr-c ...	Expand or contract the scale of both the backwards space-time patterns, and the nodes of the attractor basins showm in decimal, hex or as a bit pattern, see section 26.2.
ppstack-p ...	Toggle the pause on and off when showing the partial pre-image histogram for networks with non-local wiring, such as RBN. A top right reminder is also presented, **partial pre-image histogram: start pause-p:** See section 29.9.2 for further details.

30.4 Attractor basin complete options

When the attractor basin is complete, or if **o** was entered when interrupting in section 30.2, or if the attrctor basin was abandoned in section 30.2.4, the following options are presented in a top left window,

> **graphics-g image: prt/save/load-p/s/l, ops-o**
> **toggle: single-field-t STP-P, seed-e** (seed-e *not for the basin of attraction field)*
> **net-n, back-q cont-ret:**

Enter **q** to backtrack to previous prompts. Enter **return** to generate another attractor basin, for a mutated network according to the mutations set in chapter 28, all other parameters and settings remaining unchanged.

The other options are summarised below,

option ...	*what it means*
graphics-g ...	enter **g** to alter the graphics setup described in section 6.3.
image: prt/save/load -p/s/l ...	enter **p**, **s** or **l** to print, save, or load the DDLab screen image but without the top left prompt window in this section. For printing see section 5.5, for saving and loading see chapter 35. Printing, saving, and loading the DDLab screen can also be done in the usual way (but showing the screen as is), by entering the following at any time when the prompt cursor is flashing (see section 5.4), **@** - to print > - to save < - to load
ops-o ...	enter **o** to open up a range of further rule, network and seed options, which are presented in a top right window. These options are described in section 30.5 below.
single-field-t ...	(or **field-single-t**) enter **t** to toggle between a single basin and the basin of attraction field. The prompt sequence will revert to selecting the seed (see chapter 21) or the the first presentation parameters prompt (section 24.1). Note that the "layout" defaults (see chapted 25) may need to be reset.
STP-P ...	enter **P** to toggle between showing and not showing "backwards" space-time patterns (see section 24.7).
seed-e ...	enter **e** to revise the seed for a single basin or subtree. The seed options described in chapter are 21 presented in a lower central window. This option does not appear for a basin of attraction field.
net-n ...	enter **n** for the many network architecture options described in chapter 17, including viewing, revising and filing, the Derrida plot (chapter 22, and learning (chapter 34), a repeat of the option in section 30.4.

If one of these option is selected, the program eventualy returns to the options prompt above.

30.5 Further attractor basin complete options

Enter **o** in section 30.4 above to display a range of further (context dependent) options for amending the rule, the network, or seed (for a single basin or subtree onle). The following is presented in a top right window,

> **rule:save/load/revise/rnd/trans-s/l/v/r/t, net-n**
> **bitflip-1, flip ends-w/b, Z:higher/lower-Z/z** *(***Z:***... for single rule networks only)*
> **seed:rev-e rnd/blk-R/k sng:pos/neg-5/6 save/load-S/L**
> **goback-q cont-ret:** *(***seed:***... not for the basin of attraction field)*

Enter **q** to backtrack to previous prompts in section 30.4.

Enter **return** to generate another attractor basin, for a mutated network according to the mutations set (in chapter 28), which are only put into effect if the rule and seed parameters are not revised with the options below. The remaining options are listed in two categories, "revising rule/s" (section 30.5.1), and "revising the seed" (section 30.5.2), for single basins or subtrees only. Changes to rules or seeds generaly result in new attractor basins being started. However, changes made during an interrupt when attractor basins are incomplete, and continue from the point of interrupt, will result in errors as described in sections 30.2.3.

30.5.1 Attractor basin - revising rule/s

The meaning of the options in section 30.5,

> **rule:save/load/revise/rnd/trans-s/l/v/r/t, net-n**
> **bitflip-1, flip ends-w/b, Z:higher/lower-Z/z**

for revising the rules, or rules in mixed rule networks, are sumerised below,

option ...	*what it means*
save/load-s/l ...	enter **s** or **l** to save or load the rule as a `.rul` file (see Filing, chapter 35). For mixed rule networks this will apply to the rule at cell index 0 only.
revise-v ...	enter **v** to revise the rule. For mixed rule networks the following prompt is presented below the top right window to select the cell index, **enter cell index, 9-0:** *(for example)* Then the rule is selected in a lower right window as described in chapter 16.
rnd-r ...	enter **r** to reset a new random rule, or all rules at random in a rule mix.
trans-t ...	enter **t** to transform the the rule as described in chapter 18. For mixed rule networks this will apply to the rule at cell index 0 only (section 18.7), but Canalizing inputs can be set for the whole network (chapter 15).

net-n ... enter **n** for the many network architecture options described in chapter 17, including viewing, revising and filing, the Derrida plot (chapter 22, and learning (chapter 34).

bitflip-1 ... enter **1** for a random bit-flip in the rule-table, or for all rule-tables in a rule-mix.

flip ends-w/b ... enter **w** or **b** to flip the output of the all 1s or 0s neighborhood in a single rule, or for all rules in a rule mix.

Z:higher/lower-Z/z ... for single rule networks only, enter **Z** to progressively force the Z-parameter higher, or enter **z** to force it lower, as in section 16.9.3.

Once the rule is revised it will be shown in the lower rule window described in section 16.14.

30.5.2 Attractor basin - revising the seed

for a single basin or subtree only

The meaning of the options in section 30.5,

seed:rev-e rand/blk-R/k sng:pos/neg-5/6 save/load-S/L

for revising the seed, which apply for a single basin or subtree only, not for a basin of attraction field, are sumerised below,

option ... *what it means*

rev-e ... enter **e** to revise the seed for a single basin or subtree. The seed options, described in chapter are 21, are presented in a lower center window, a repeat of the option in section 30.4.

rand/blk-R/k ... enter **R** for a new random seed, and **k** for a random central block of cells, the remainder set to 0. The block size was specified in section section 21.4.

sng:pos/neg-5/6 ... enter **5** for a positive singleton seed, a central cell set to 1, the remainder set to 0. Enter **6** for a negative singleton seed.

save-S ... enter **S** to save the seed or the current state. The top right following prompt is presented,

save: original seed-s, current state-c:

This will be saved as a `.eed` file, (see Filing, chapter 35).

load-L... enter **L** to load a new seed, as a `.eed` file, (see Filing, chapter 35).

Chapter 31

Output parameters for space-time patterns

This chapter describes output parameter options that allow various default settings to be revised before space-time patterns start. Many of these options can also be invoked on-the-fly (chapter 32, and also from the interrupt window (section 32.14). Although there are many output parameter options and sub-options, all have defaults. The options may be skipped unless particular defaults need to be changed, and its possible to jump directly to a particular category of options, namely *analysis* and *histograms*, apart from the first set of of miscellaneous options.

The options apply only if the network was first set up to run "forward only" and see just space-time patterns. This means that a "single basin/subtree" should have been selected at the first DDLab prompt in section 6.1 by entering **s** (a seed would also have been set). Also, "space-time patterns only" should have been selected at the the first output parameter prompt for attractor basins in section 24.1, or alternatively, selected at a final stage at the single basin prompt, section 29.3 (enter **s** in both cases).

31.1 The first output parameter prompt for space-time patterns

If **s** is selected in section 24.1 or section 29.3, the first output parameter prompt for attractor basins is presented in a top right window as follows,

FORWARD ONLY options: analysis-a histograms-h (ALL-def):

Enter **d** to accept all output parameter defaults and skip these options. This can be done at any stage in the output parameter prompt sequence to accept remaining defaults and skip remaining options. A reminder to this effect **accept defaults-d** remains displayed in a top center window.

All the output parameter options may be looked at in turn, but they are divided into four categories to allow jumping directly to the category where options need to be set. The categories are,

revise from start-ret ... various miscellaneous options difficult to categorize: sequential updating, cell color by neighborhood or value, options for pausing or stepping through space-time patterns, displaying space-time patterns in other than the "native" dimension, scrolling, and probabilistic networks.

analysis-a ... various methods of viewing and analyzing space-time patterns, including pattern density, input-frequency/entropy, applied to the whole network or one chosen cell. The size of the window of time-steps for these analyses, and the threshold for 'frozen" cells, can be amended.
Plots of the "state-space matrix", and the "return map".

histograms-h ... various statistical methods of analyzing space-time patterns, where the data is displayed as a histogram, including damage spread from perturbations (1d and 2d only), and data on attractors and "skeletons".

Enter **a** to skip directly to the *analysis* category of options (section 31.8), or enter **h** for the *histogram* category (section 31.14).

Enter **return** for the first set of miscellaneous options, described below,

---

31.2 Sequential updating, space-time patterns

By default, network updating is synchronous, in parallel. Alternatively, the updating can be sequential. There are $n!$ possible sequential updating orders in a network of size n. DDLab provides simple default orders, forward or backward along the network index, or a random order. For $n \leq 31$, any specific order can also be set, filed and loaded (from a `.ord` file). These options are the same as in section 29.7 for attractor basins, so are not described in detail here. Note that parallel and sequential updating (whatever order was set) can be toggled on-the-fly with **U**.

If **return** is entered in section 31.1, the following top right prompt is presented,

sequential updating-s, parallel-def:

Enter **s** to select sequential updating. The following top right prompt is presented to specify the order,

for $n > 31$
sequential updating:->(def) <-b rand-r:
for $n \leq 31$
sequential updating:->(def) <-b rand-r set-s all-S:

For $n > 31$, usually the case when studying space-time patterns, the sequential order can be specified as the default left to right (enter **return**) or the opposite (backward) order, from right to left (enter **b**), see section 29.7.1 for details. Enter **r** for a random order, see section 29.7.2 for details.

For $n \leq 31$, a specific order can be set (enter **s**), see section 29.7.3 for details. To list all possible orders (enter **S**), an order can be copied from the list, see section 29.7.4 for details.

31.3 Color cells by value or neighborhood

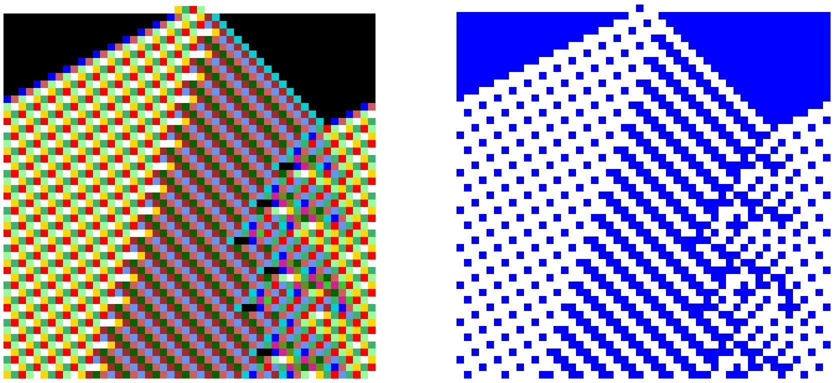


Figure 31.1: 1d space-time pattern colors, 50 time-steps for a CA n=50, $k = 5$, rule 82 26 dc 23. Space is across, time is down the page.
left: by neighborhood, the default.
right: by cell value.

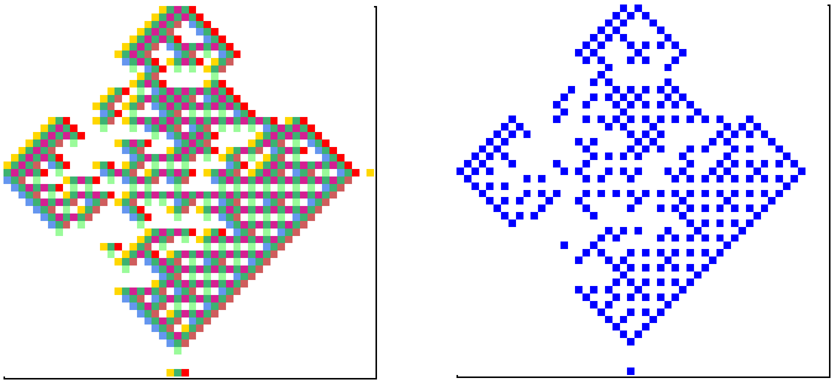


Figure 31.2: 2d space-time pattern colors, the 50th time-step for a CA n=50×50, $k = 5$, rule 1a 1b c1 26.
left: by neighborhood, the default.
right: by cell value.

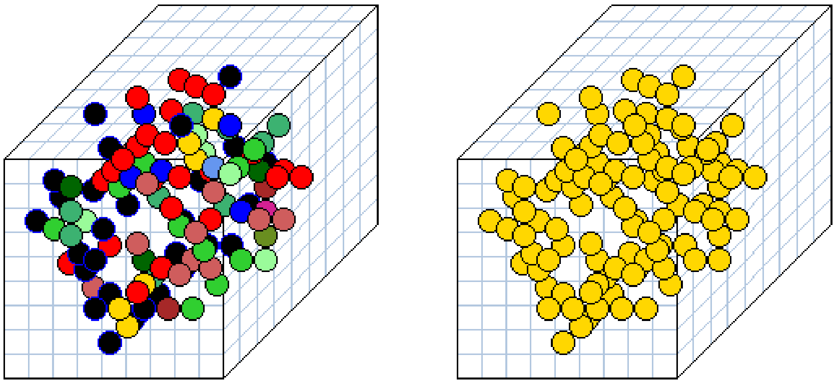


Figure 31.3: 3d space-time pattern colors. The 5th time-step for a CA n=9×9×9, $k = 7$.
left: by neighborhood, the default.
right: by cell value.
In both cases cells only cells with value 1 are displayed.

By default each cell is colored according to the rule-table index that was "looked up" to determine the cell's value, in other words the cell's neighborhood at the previous time-step. These colors are assigned from a palette of 16 (see section 6.4.4 for the color code). Alternatively cells may be colored simply according to their value, 0 or 1. Figures 31.1, 31.2 and 31.3 show the difference between neighborhood and value display of space-time patterns for 1d, 2d and 3d (these were all generated form a single central 1 seed).

To select color by value, the following prompt is presented in a top right window,

color cells by value-v, by nhood-def:

Enter **v** to color cells by value, or accept the default, by neighborhood. Note that this can be toggled on-the-fly (see section 32.8.3). Other color presentations that depend on "frozen" cells, the frequency of 1s in a time window, and filtering space-time patterns, can also be toggled on-the-fly (see section 32.10).

31.4 Pause and step

DDLab offers various methods of scanning space-time patterns, some of which are available on-the-fly (see section 32.2), and some from the interrupt window (see section 32.14). From the output parameters, options are offered to make single time-steps, to pause after a given number of time-steps, or to always pause when 1d (or 2d isometric) space-time patterns reach the foot of the screen. No pause is the default.

The following top right prompt is presented,

step-s, pause screen full-p, or enter time-step (no pause-def):

Enter **s** to show each next iteration by pressing **return**. Enter **p** to pause when the screen is full.

Alternatively, enter a number to specify a run of time-steps before a pause. At the pause the interrupt options window will appear (see section 32.14), enter **x** for the next run of time-steps.

31.5 Space-time patters in other than the native dimension

Space-time pattern can be displayed in a dimension other than the "native" dimension. A 1d network can be displayed in 2d or 3d, a 2d in 1d or 3d, and 3d in 1d or 2d. This can also be changed on-the-fly in section 32.9.1. The following top right prompt is presented,

for 1d
1d network, show STP in 1d-1 2d-2 3d-3 (def 1d):
for 2d
2d network, show STP in 1d-1 2d-2 3d-3 (def 2d):
for 3d
3d network, show STP in 1d-1 2d-2 3d-3 (def 3d):

If not the native dimension, the 2d and 3d axes i, j, h are set automatically by factorizing n. A native 3d network will be displayed in 2d as a stack of horizontal 2d slices as in figure 32.9.1.

Note that if n is prime, i will equal n, and both j and h will equal 1. Then if 2d or 3d is selected, a single row of cells will be displayed across the top of the screen in the 2d or 3d format.

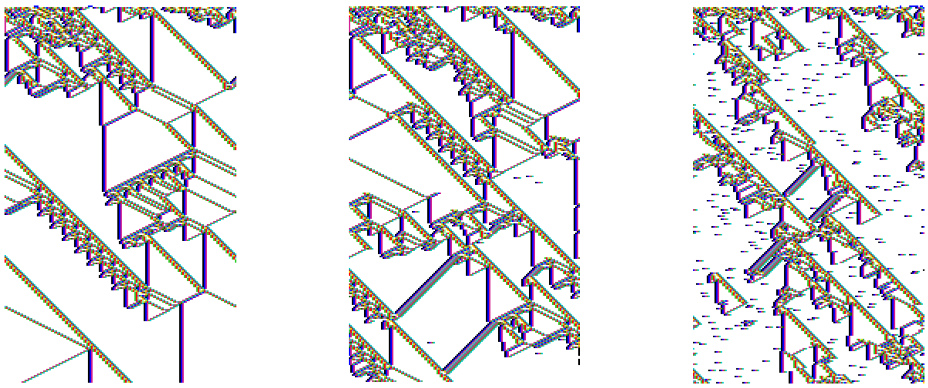
31.6 Scrolling 1d space-time patterns

For 1d networks displayed in 1d (see section 31.5 above), the following prompt allows space-time patterns to be scrolled upwards rather than displayed in successive vertical sweeps.

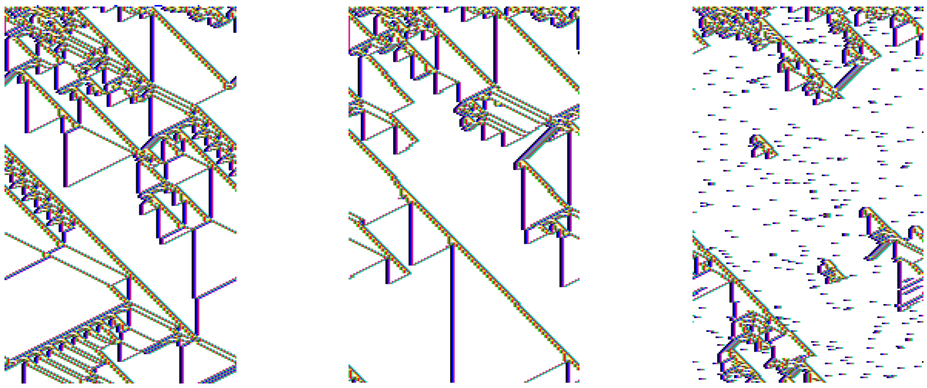
scroll space-time pattern-s:

Enter **s** to scroll. The two methods can also be toggled on-the-fly (see section 32.12.3). Note that if the input-entropy or pattern density is set in section 31.9, these plots will also be scrolled.

31.7 Probabilistic networks



default prob=100% output prob=99.8% output prob=98%



update prob=99.8% update prob=98% output+update=98%

Figure 31.4: Probabilistic networks. The "update" probability that a cell will be updated at all, and the "output" probability updating will be according to the rule-table (otherwise at random), see "probabilistic networks" (section 31.7). These examples are for the k=5 CA rule e9 f6 a8 15, n=150, 245 time-steps from the same initial state. Cells colors are by neighborhood (see section 31.3) and filtered to emphasize gliders (see section 32.10).

There are two types of probabilistic networks where the probability, by default 100%, can be reset to a lower value.

Firstly the "update" probability, that a cell will be updated at all, gives some asynchronicity in the dynamics. If this probability is set to P, there is a $1 - P$ chance that a cell is not updated, thus remaining the same at the next time-step.

Secondly the "output" probability, that a cell is updated according to the rule-table, which introduces noise. If the probability is set to P, there is a $1 - P$ chance that a cell's value will be randomly assigned.

Figure 31.4 gives examples. The following top right prompt is presented,

set probability (def 100%) 0-100, update: output:

First set the the required update probability, then the output probability. These are set as a percentage, which may be a decimal number.

If *both* probabilities are set to less than 100%, then the "update" probability will be implemented first, and only cells that were updated normally will be subject to "output" probability.

31.8 The "analysis" category of options

Options from section 31.9 to section 31.13 belong to the *analysis* category, somewhat arbitrary named, which describes various methods to analyze space-time patterns.

The methods are provided in this category include,

- The pattern density.
- Input-frequency and input-entropy.
- The state-space matrix.
- The return map.

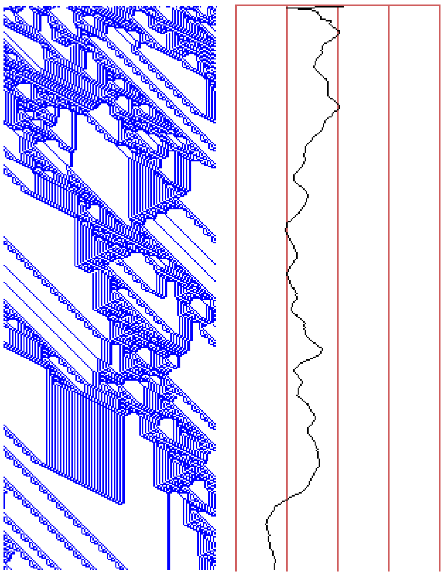
Enter **a** in section 31.1 to skip directly to the first "analysis" option, section 31.9, or arrive here by looking at each output parameter in turn.

Further methods are in the *histograms* category of prompts (section 31.14). Enter **h** in section 31.1 to skip directly to the first histogram option.

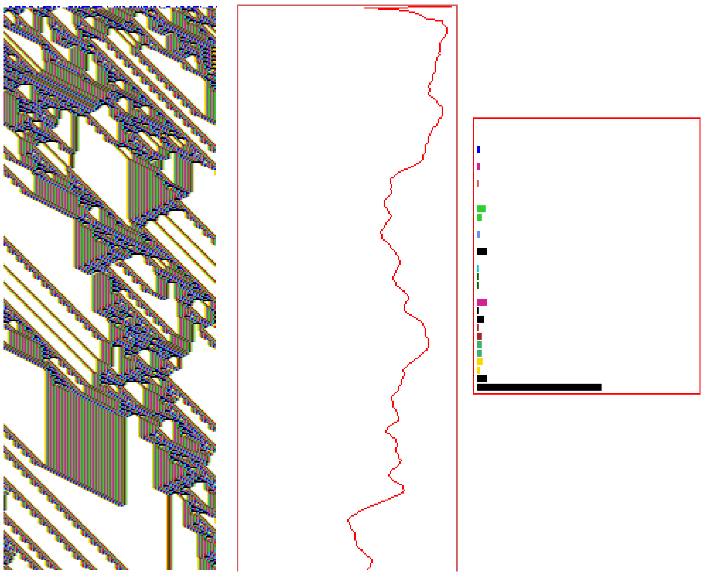
31.9 Input-entropy and pattern density

The first two methods provided in the "analysis" category are,

- The pattern density (the density of 1s at each time-step).
- The look-up frequency of neighborhoods in the rule-table (input-frequency) and the resulting "input-entropy".



Density plot: A 1d space-time pattern (color by value), with a density plot alongside. The x axis shows the density of 1s. Vertical divisions (from the left) mark densities of 0, 0.25, 0.5, 0.75 and 1. Time is on the y axis.



Input-Entropy plot: The same 1d space-time pattern (color by neighborhood). To the extreme right is the input frequency histogram, with the neighborhood all 1's top, and all 0's bottom. The input-entropy, the entropy of this histogram is shown center, on the x axis, with zero entropy on the left and maximum entropy on the right. Time is on the y axis.

Figure 31.5: The pattern density plot (left), and input-entropy plot and input frequency histogram (right), for the k=5 1d CA 6e ee 93 60, n-150, 40 time steps from the same random initial state, measured over a moving window of 10 time-steps.

These measures are displayed as a graph alongside the space-time patterns. The input-frequency is simultaneously displayed as a histogram, which may be expanded into a 3d histogram, with an extra time axis (see figures 31.5 and 31.6 and the on-the-fly options in section 32.11.3). A further on-the-fly toggle option (section 32.11.2) shows superimposed graphs of the input frequency for each neighborhood, with or without the input-entropy plot.

The following top right prompt is presented,

for 1d networks with homogeneous k
ANALYSIS: none-n, pattern density-1,
input-frequency/entropy: histogram-(def), +time-3:

for 2d and 3d networks, or for mixed k
ANALYSIS: none-(def), pattern density-1,
input-frequency/entropy: histogram-2, +time-3:

The default for 1d networks with homogeneous k is to display a plot of the input-entropy, and the input-frequency as a histogram. For 2d or 3d networks, or networks with mixed k, the default is *not* to display either analysis, Its possible also to restrict these methods to selected single cells rather than the entire network. For input entropy in mixed k networks this is the only option.

If input-entropy and pattern density is set, the display can be toggled on-the-fly between the two (see section 32.11.1), and a scatter plot of the two measures can be displayed where rules have characteristic signatures (see figure 32.14 and section 32.11.4).

Further on-the-fly options plot the mean entropy against the standard deviation of the entropy for a sample of rules, giving a 3d histogram that separates out order, complexity and chaos (see chapter 33).

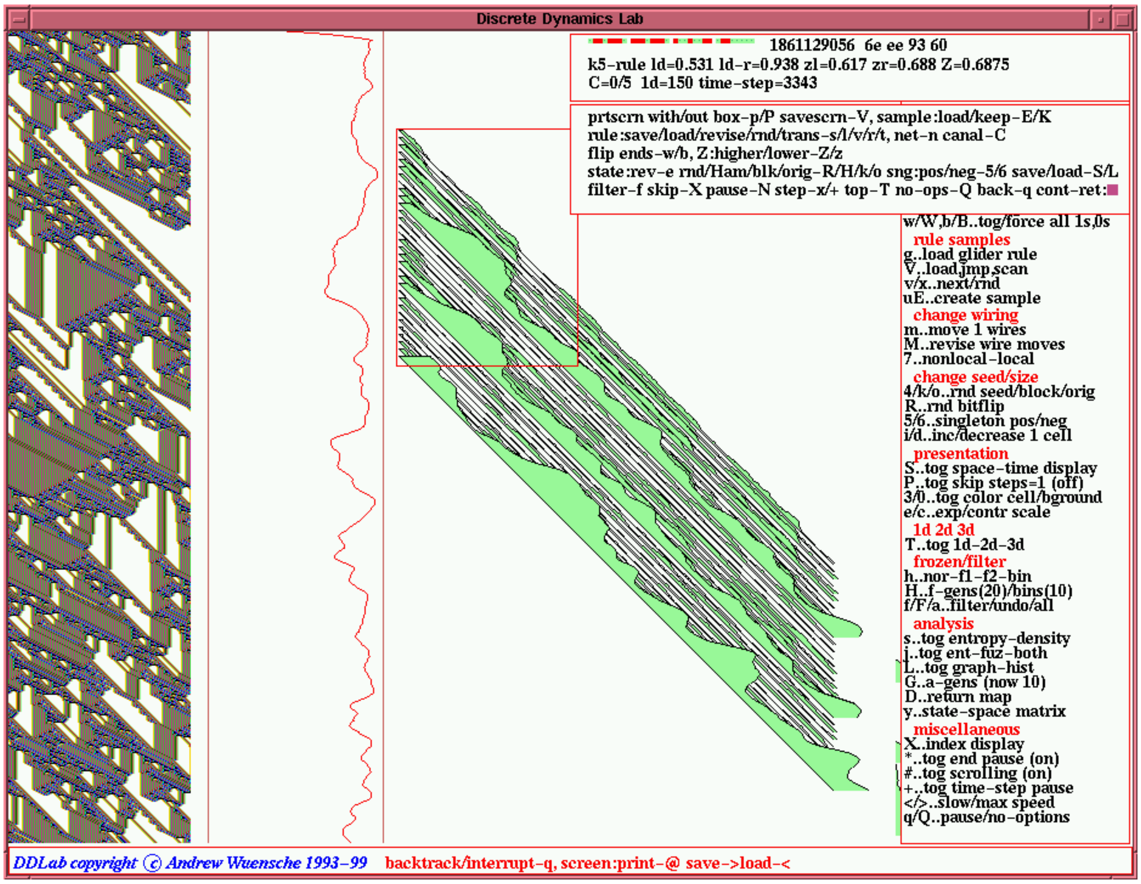


Figure 31.6: The input frequency histogram in 3d. The DDLab screen showing: (*left)* The space-time patterns for the k=5 1d CA 6e ee 93 60, n-150 (color by neighborhood). (*center left)* The input-entropy plot, which was subsequently toggled to show a superimposed graphs of the input frequency, or both plots together (see section 32.11.2). (*center)* The input frequency histogram in 3d. This can be preset, or toggled on-the-fly between 2d (as in figure 31.5 and 3d. (*right)* The key to on-the-fly options (see sections 32.1 - 32.2). (*top right)* The options window that appears when space-time patterns are interrupted.

31.9.1 Pattern density

Enter **1** in section 31.9 above at to show the pattern density. The horizontal scale represents the density of 1s at each time-step, or for the last x time-steps. The size of this moving window of time-steps can be reset (see section 31.11 below). The vertical time axis corresponds to each time-step as drawn.

31.9.2 Input-frequency and input-entropy

Enter **return** (or **2** for 2d or 3d networks) in section 31.9 to show a 2d histogram of the input-frequency (on its side), with 2^k columns representing the lookup frequency of each rule-table entry, and colored accordingly. If **3** is entered in section 31.9, the histogram will be expanded to 3d, with an extra time axis, shown as an axonometric projection, as in figure 31.6. An on-the-fly option allows toggling between the 2d and 3d histogram (see section 32.11.3).

The entropy of the histogram frequency distribution, the input-entropy, will be displayed to the right of the space-time pattern. Its possible to toggle on-the-fly between between a display of the input-entropy, superimposed graphs of the input frequency of each neighborhood, and both of these plots shown simultaneously (see figure 31.6 and section section 32.11.4).

31.10 Single cell input-entropy and pattern density

All the analysis options and methods of presentation described in section 31.9 can be applied to just one selected singe cell. This is usually done over a much larger moving window of time-steps, the default in 500.

In a mixed k network, if input-entropy was selected (if **2** or **3** was entered in section 31.9), if a single cell position is not specified, it will be applied automatically, usually at position 0.

In general, the following top right prompt is presented,

single-s all-(def)

If **s** is selected, the position of the single cell needs to be specified (in 1d, 2d or 3d) with the following top right prompt (for a mixed k network with input-entropy selected, this will be the first prompt),

for a 1d network n=150
single: enter index (max1149 def 0):

for a 2d network 40× 0
single 2d: 2d index (max 39,39, now 0,0)
enter i: enter j:

for a 3d network 22×22×22
single 3d: 3d index (max 21,21,21, now 0,0,0)
enter i: enter j: enter h:

When the space-time pattern is running, a reminder of the single cell position (and the generation size, see section 31.11 below) will appear in a top right window, for example in a 3d network,

single entropy: pos i,j,h=11,12,13 gens=500

A similar reminder also appears in the "on-the-fly key index" (see section 32.11.5).

31.11 Analysis generation size

The analysis of space-time patterns set in section 31.9 relates to a moving window of time-steps, or generations. For input-entropy these are past time-steps, for pattern density the present time-step is also included.

The default analysis generation size is 10, unless a single cell was selected in section 31.9, in which case the default size is 500. These values can be reset, new setting becomes the default. The following top right prompt is presented, for example,

enter new 'analysis' generation size (now 10):

The analysis generation size can also be reset on-the-fly, and is shown in the "on-the-fly key index", together with the single cell position, if selected in section 31.10.

31.12 Frozen generation size

Once running, the space-time pattern presentation can be reset on-the-fly to highlight "frozen" cells in a number of ways. These are cells that have not changed in the previous x time-steps (see section 32.10.1). The number of time-steps, or generations, making up the frozen threshold may be reset from the initial default of 20. The new setting becomes the default. The following prompt is presented,

enter new 'frozen' generation size (now 10):

The frozen generation size can also be reset on-the-fly, and is shown in the "on-the-fly key index" (see section 32.10.2).

31.13 State-space matrix and return map

As in section 24.2 for attractor basins, the state-space matrix plots each state in the space-time pattern on a 2d grid near the bottom of the screen, plotting the left half of the bit string against the right half. The y-axis represents the right $n/2$ bits. If n is odd, the extra bit is included on the left, and the grid is a flat rectangle, otherwise the grid is square.

In the return map, each state (bitstring), length n, $B_0, B_1, B_2, B_3 \ldots B_{n-1}$ is converted into a decimal number r, 0-2, where,

$$B_0 + B_1/2 + B_2/4 + B_3/8 + \ldots + B_{n-1}/2^{n-1}$$

As the network is iterated, this state value at time-step t (x-axis) is plotted against the state value at timestep t+1 (y-axis) in the bottom center of the screen. From a classical dynamical systems viewpoint, note the fractal structure of the resulting trajectories and attractors.

The following prompt is presented,

show state-space matrix-m, return map-r:

Enter **m** to show the state-space matrix, **r** to show the return map. Both options can also be toggled on-the-fly (see sections 32.11.6 and 32.11.7).

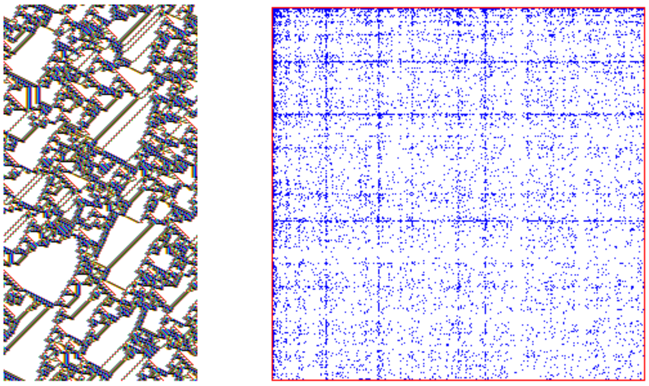


Figure 31.7: The state-space matrix for the $k = 5$ rule 7a 6e 69 84, n=150, plotting the left half against the right half of each state bit string, for about 10,000 time-steps.
left: an example of the space-time pattern, color by cell value, and filtered to show gliders.
right: the state-space matrix.

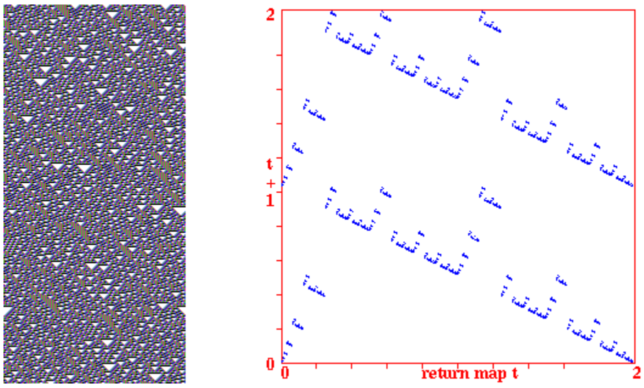


Figure 31.8: The return map for the $k = 3$ rule 30, n=150, plotting the state at each time-step, converted into a decimal number 0-2, against its successor, for about 10,000 time-steps.
left: an example of the space-time pattern, cells colored by value.
right: the return map, note the fractal structure.

31.14 The "histograms" category of options

Options from section 31.15 to 31.19.5 belong to the *histogram* category, with further, mainly statistical, methods for analyzing space-time patterns, including,

- The difference between two networks, or "damage".
- Statistical data on attractors types, and "skeletons", suitable for very large networks.

Note that to considerably speed up computation for generating these histograms, the display of space-time pattern graphics can be toggled off on-the-fly with key **S**, see section 32.8.

Enter **h** in section 31.1 to skip directly to the first "histograms' option, section 31.15 below, or arrive here by looking at each output parameter in turn.

31.15 Damage, the difference between two networks

not applicable to 3d

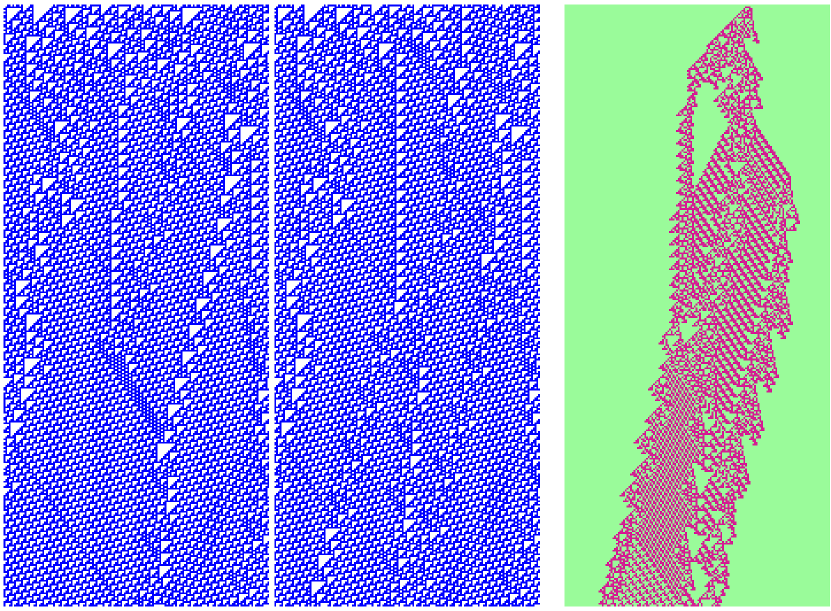


Figure 31.9: The difference in the dynamics between two 1d CA. *left*: The space-time patterns of two identical CA (color by value), except that the random initial state differs by one bit. *right*: The difference between the two at each time-step shown as a third space-time pattern. k=3 rule 110, n=150.

Its possible to graphically represent the difference, or the spread of "damage", between the space-time patterns of two networks. Typically the networks are identical except for a 1 bit difference in their initial state, and the damage is shown as a "purple stain" on a third space-time pattern, spreading from the different bit. This option works for 1d and 2d networks only.

Damaged cells can be defined in two ways. Once damaged, keep the damage irrespective of future dynamics, or show simply the difference at each time-step.

Methods are also provided for automatically gathering statistical data on the sizes of damage spread from many random initial states that differ by one bit, shown as a histogram. This is a method of analyzing order and chaos in networks, especially in random Boolean networks applied as idealized models of genetic regulatory networks[4, 17]. The following top right prompt is presented,

HISTOGRAMS: show difference between 2 networks
damage: keep-f, dont keep-F, show stats-s:

Enter **s** to shown as a histogram of damage spread for a sample of initial states generated automatically.

Otherwise, for a manual version of damage spread, not showing a histogram, enter **f** to keep the damage, or **F** to show just the difference at each time-step.

31.15.1 Duplicate the network and seed

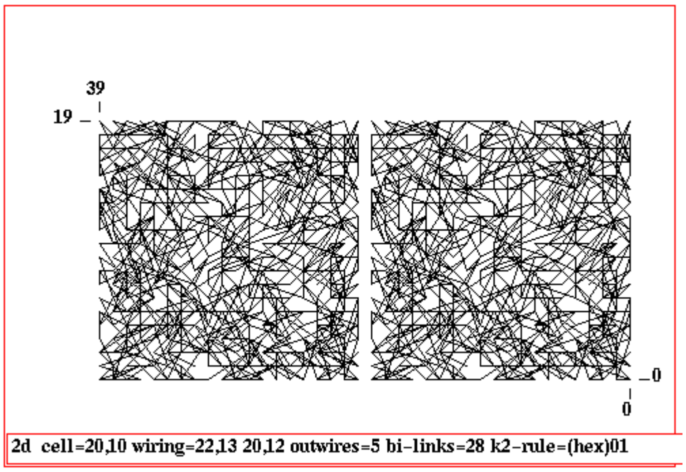


Figure 31.10: A duplicated 2d RBN, 40×20, k=2, showing just all the links. The network consists of two identical 20×20 sub-networks

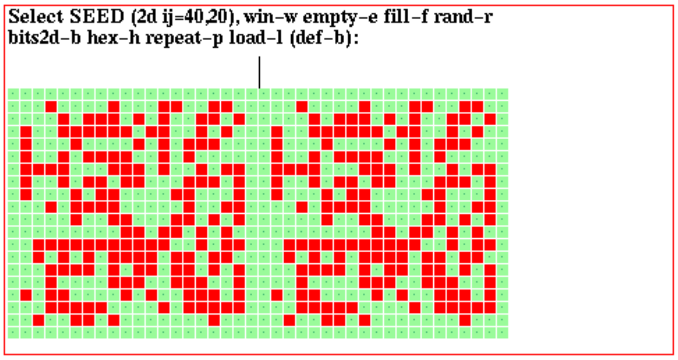


Figure 31.11: A duplicated 2d seed, 40×20. consisting of two identical 20×20 sub-seeds

A subsequent option, normally selected together with the "damage" option (section 31.15) above, duplicates both the network and the initial state. This actually doubles the size of the original network, making a network containing two identical sub-networks, each with the same initial state. The following top right prompt is presented,

duplicate network and seed-y:

Enter **y** to duplicate the network. Though normally used together, the 'damage" option (section 31.15) and this option can be used independently.

If "damage" was selected and the network is not duplicated, the original network and its initial state will simply be notionaly divided in two, and the "damage" displayed will be between the two halves of the network. However, the network can only be divided if n for 1d, or i for 2d, is even, otherwise the following top right message is displayed,

cant divide odd width network, cont-ret:

Once the network is duplicated, prompts are presented to look at and possibly modify the new network and the seed. Firstly the "network architecture" prompt described in section 17.1, secondly the seed prompt in section 21.1.

When the network is run with these option set, a third space-time pattern shows the difference between the two sub-networks as in figures 31.9 and 31.15.1.

Note that if **f** or **F**, for a manual version of damage spread, was selected in section 31.15, and the network was duplicated in this section, then there will be no difference in the initial states, thus no damage. To introduce a difference, a random bitflip can be made on-the-fly (with **R**, see section 32.7.2). A new random initial state (the same for each sub-network) can also be set on-the-fly (with **4**, see section 32.7.1).

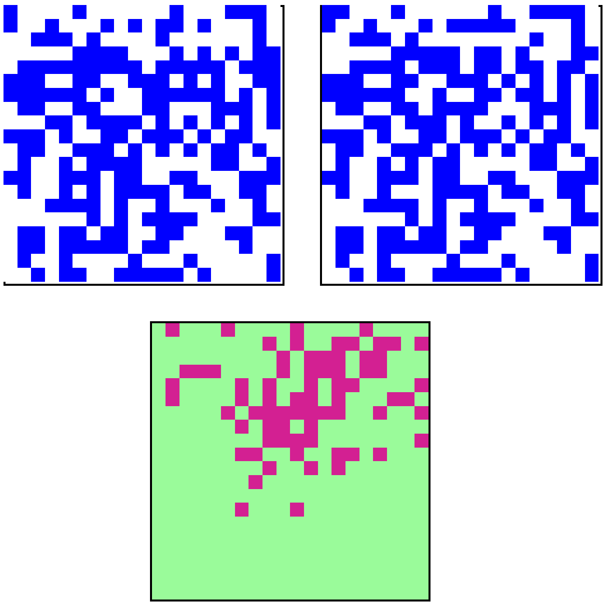


Figure 31.12: An example of 2d "damage". The difference in the dynamics between two 2d RBN. *top*: The space-time patterns of two identical 20×20 RBN (color by value), except that one bit was flipped in the the initial state. The RBN has k=2, with connections as in in figure 31.15.1. *bottom*: The "damage spread" between the two, which has stabilized after a number of time-steps, shown as a third space-time pattern. In this example **keep-f** was selected in section 31.15, so that once a cell is "damaged" - it stays "damaged".

31.16 The Damage Histogram

not applicable if $n < 16$ or 3d networks

If **s** (for statistics) is entered in section 31.15, a histogram of damage spread for a sample of initial states will be generated automatically. First the prompt to duplicate the network and seed in section 31.15.1 is presented. Dont duplicate if this was done previously, otherwise the duplicated network will be reduplicated, making 4 copies of the original.

A sequence of top right prompts are presented to set various parameters,

Damage Spread: show binned damage-b, actual sizes-def:
if **return** *is entered...*
select cut-off damage size (min 20, def 400): *(for sub-networks 20×20)*
if **b** *is entered...*

select no of damage bins (min 10 def 100):
followed by...
delay damage, time-steps before damage start (def 0):
define damage, time-steps same damage (def 5):

The "delay damage" is the number of time-steps before the two sub-networks are compared and the measure of the damage starts. This can be changed this from the default 0.

"Define damage" defines at what point the damage is deemed to have stopped spreading. By default, if the damage has not changed for 5 consecutive time-steps, it is considered to be complete - to have stabilized, but this number can be altered.

The "actual" and "binned" options are described in sections 31.16.1 and 31.16.2 below.

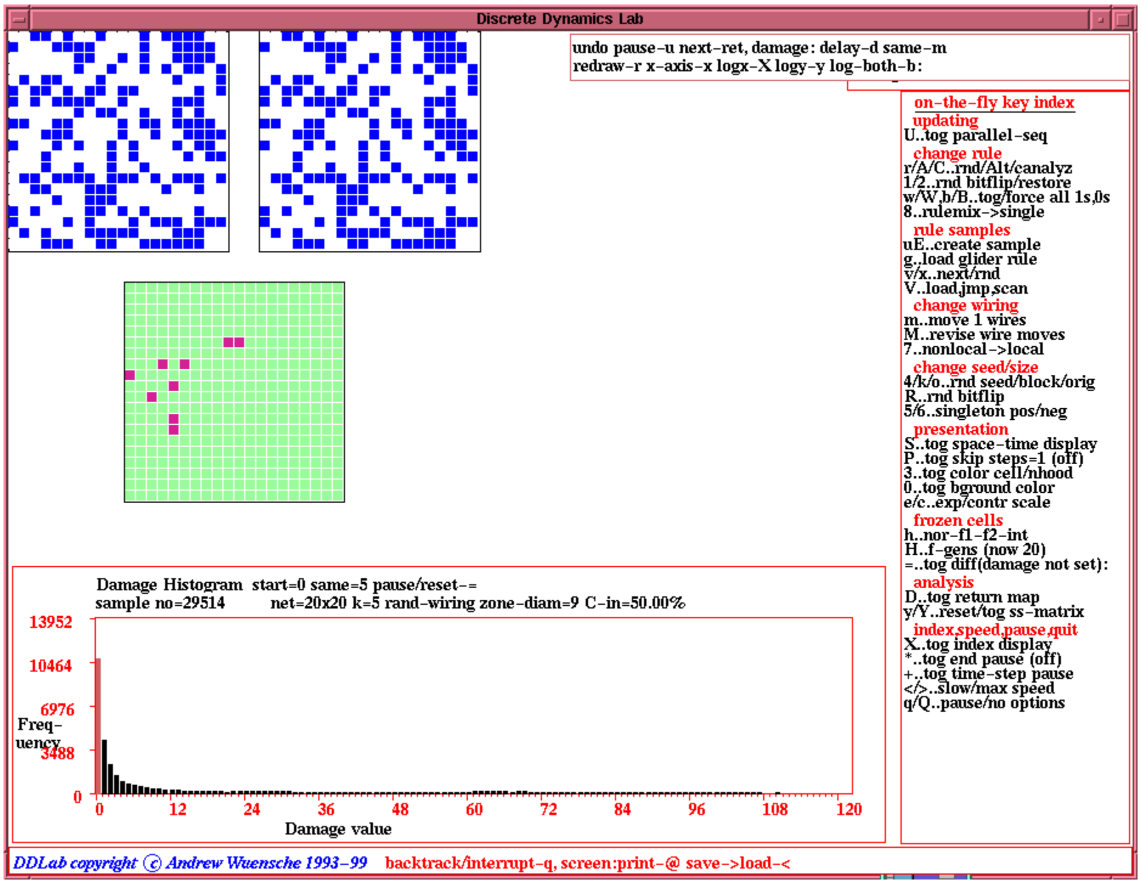


Figure 31.13: The DDLab screen, showing the automatic process for gathering statistics on damage. This example is for two 20×20 RBN sub-networks, k=5, with canalizing inputs set to 50%, see chapter 15). At each pass, a new random initial state is assigned to the sub-networks, differing by one bit at a random position. When the damage stops spreading, the histogram of the damage size (x axis) against the frequency of damage size (y axis), is updated.

31.16.1 Actual damage sizes

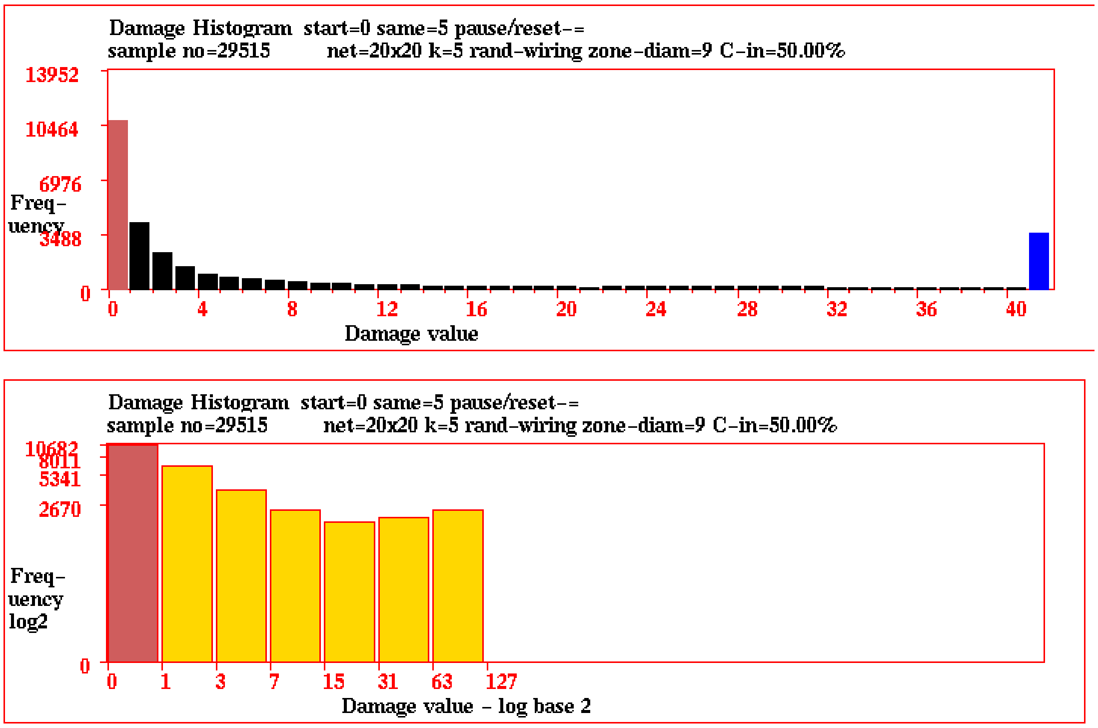


Figure 31.14: The damage histogram showing actual damage, for the same network as in figure 31.13. *top:* With the damage cut-off set at 40 (see above). The frequency of damage of 41 and above is represented by the blue bar on the extreme right. *bottom:* The the damage histogram as a log-log plot.

Enter **return** at the first damage histogram prompt (section 31.16) to present the histogram according to the actual damage sizes. This option will also allow the axes of the histogram to be subsequently rescaled, and shown as a log-log plot. The next prompt sets the cut-off size of the damage, where any damage above the cut-off is lumped together in one excess bin.

select cut-off damage size (min 20, def 400): *(for sub-networks* 20×20*)*

When the histogram is being drawn (see section 31.16.3), enter = (the equals sign) to pause, the following top right prompt is presented to amend parameters and presentation,

undo pause-u save/load-s/l next-ret, damage: delay-d same-m
redraw-r x-axis-x logx-X logy-y log-both-b:

These prompts are explained below,

undo pause-u ... enter **u** to continue the histogram without pause.

save/load-l/s ... enter **s** to save the historam data as a `.his` file (see Filing, chapter 35). Enter **l** to load the data and see it in the Xterm window (not for DOS). The file holds the frequencies (0 to the maximum current x axis) in successive pairs of bytes.

next-ret ... enter **return** to continue the histogram, but pause after the next sample.
delay-d ... enter **d** to alter the damage delay, described in section 31.16.
same-m ... enter **m** to define damage, the number of time-steps the damage must stay the same, described in section 31.16.
redraw-r ... enter **r** to redraw the histogram, which may have been overwritten by other graphics.
x-axis-x ... enter **x** to re-scale the x-axis, the following top right prompt is presented,

revise cut-off damage size, now 400 (min 20, def 400):

enter the new cut-off. Any damage above the cut-off is lumped together in one excess bin.
logx-X ... enter **x** to show the x-axis in log form, with the following top right reminder,

x-axis bined y powers of 2, cont-ret:

logy-y ... enter **y** to show the y-axis in log form, with the following top right reminder,

y axis log2, cont-ret:

log-both-b ... enter **b** to show both the x and y-axis in log form, with the following top right reminder,

y axis log2, x-axis bined y powers of 2, cont-ret:

31.16.2 Binned damage sizes

Enter **b** at the first damage histogram prompt (section 31.16) to allocate the damage to a number of equal size "bins", the next prompt sets how many,

select no of damage bins (min 10 def 100):

When the histogram is being drawn (see section 31.16.3), enter = (the equals sign) to pause,

undo pause-u save/load-s/lnext-ret

Only these options are available,

undo pause-u ... enter **u** to continue the histogram without pause.
save/load-l/s ... enter **s** to save the historam data as a `.his` file (see Filing, chapter 35). Enter **l** to load the data and see it in the Xterm window (not for DOS). The file holds the frequencies (0 to the maximum current x axis) in successive pairs of bytes.
next-ret ... enter **return** to continue the histogram, but pause after the next sample.

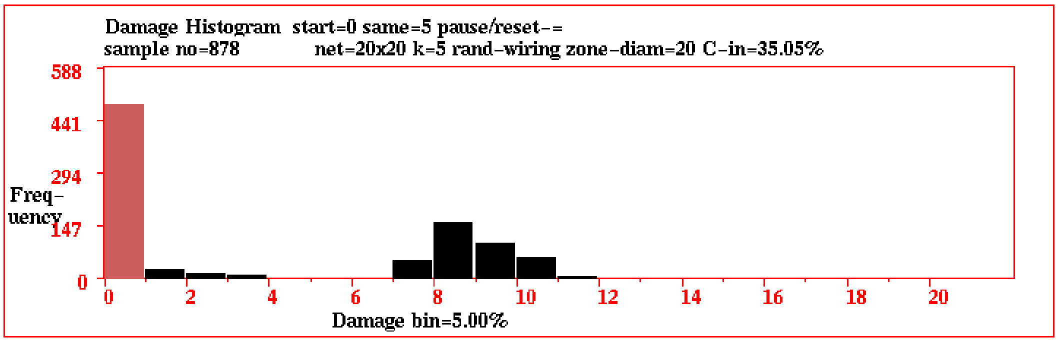


Figure 31.15: The damage histogram showing binned damage. An example of the damage histogram with the damage frequency allocated to 20 of equal size "bins". This example is for two 20×20 RBN sub-networks, k=5, (as in figure 31.13, but with canalizing inputs set to 35%. This results in less ordered dynamics and a different, bimodal, pattern of damage frequency.

31.16.3 Drawing the histogram, and pausing

The histogram is drawn in a window in the lower part of the screen, as in figure 31.13. At each pass, a new random initial state is assigned to the sub-networks, differing by one bit at a random position. When the damage stops spreading (see "define damage" in section 31.16), the histogram of the damage size (x axis) against the frequency of each size (y axis), is updated. The y axis is rescaled automatically to allow the highest histogram bar to fit.

A top right data window keeps track of the current damage, for example $\boxed{\text{damage=11/400=2.8\%}}$.

Information at the top of the histogram window, including some reminders about the network (see figure 31.14) is decoded as follows,

start= ...	the number of time-steps before the measure of the damage begins, see section 31.16.
same= ...	the number of time-steps required to define damage as having stabilized, see section 31.16.
pause/reset-= ...	reminder: enter = to pause and reset parameters.
sample no= ...	the size of the sample so far.
net=20×20 ...	network size (for example).
k= ...	neighborhood size.
local wiring ...	local wiring, as in CA
rand-wiring ...	random wiring, as in RBN
zone diam=...	the size of the relative local zone within which random wiring is confined, see section 11.5.2.
C-in= ...	the percentage of canalizing inputs.

To pause, enter enter = (the equals sign) for the histogram options described in sections 31.16.1 and 31.16.2. Alternatively, pause by entering **q** for the general pause options for space-time patterns described in section 32.13..

The space-time pattern on-the-fly options work during the histogram routine. To speed the damage histogram, toggle the space-time graphics off with the on-the-fly option **S** (see section 32.8.1.

31.17 The Histogram of Attractors or Skeletons

For large networks, especially RBN, it may be not be possible, or practical, to generate the basin of attraction field, (see Network size limitations, section 1.6).

However, statistical data on the range of attractors and their relative sizes can be obtained by running a network forwards from many random initial states, identifying different attractor "types", and creating a histogram of the frequency of each type. The data also includes the period of each attractor type, and the average transient length, the number of time-steps to reach each type. This method is suitable for networks where the attractor can actually be found in a reasonable time.

For large chaotic networks, both the typical transient length and the attractor period may be extremely (astronomically) large, making it impractical to find attractors. However, if a fraction of the network is able to stabilize (the rest remaining chaotic) as in the case of RBN models of genetic regulatory networks, a different type of quasi-attractor can be defined, called a frozen skeleton[16]. The portion of the network that must stabilize, or freeze, not change for a given number of time-steps, needs to be defined. If the stability threshold is reach, the particular pattern of stability is defined as a skeleton type. The network is run forwards from many random initial states, identifying frozen skeleton types, and making a histogram of the frequency of each type. The data also includes the average transient length, the number of time-steps to reach each skeleton type.

The option for the histogram of attractors or skeletons is presented after the damage options (sections 31.15 and 31.15.1). Enter **h** in section 31.1 to skip directly to the first "histograms" option (section 31.15), or arrive there by looking at each output parameter in turn, then enter **return** twice to skip the damage options. The following top right prompt is presented,

histogram of attractors-a skeletons-s:

31.18 Attractor histogram

Enter **a** at the prompt in section 31.17 above to generate the attractor histogram.

The next top right prompt allows the inclusion of data on initial states (seeds) that fall into the all 0s or all 1s point attractors. This is intended mainly to check the fitness of rules that have been evolved to solve the "density problem", where initial states with a density of 1s greater than 0.5 are supposed to fall into the all 1s point attractor, and less than 0.5 to the all 0s point attractor,

include seed density data for all=0/1s point attractors-a:

Enter **a** to include this data.

When the histogram is being drawn (see section 31.18.6, A top right data window gives current data, the types of attractors found, the current type, its period and average transient length, for example,

types=44 this=8 attperiod=12 avtrans=14.0

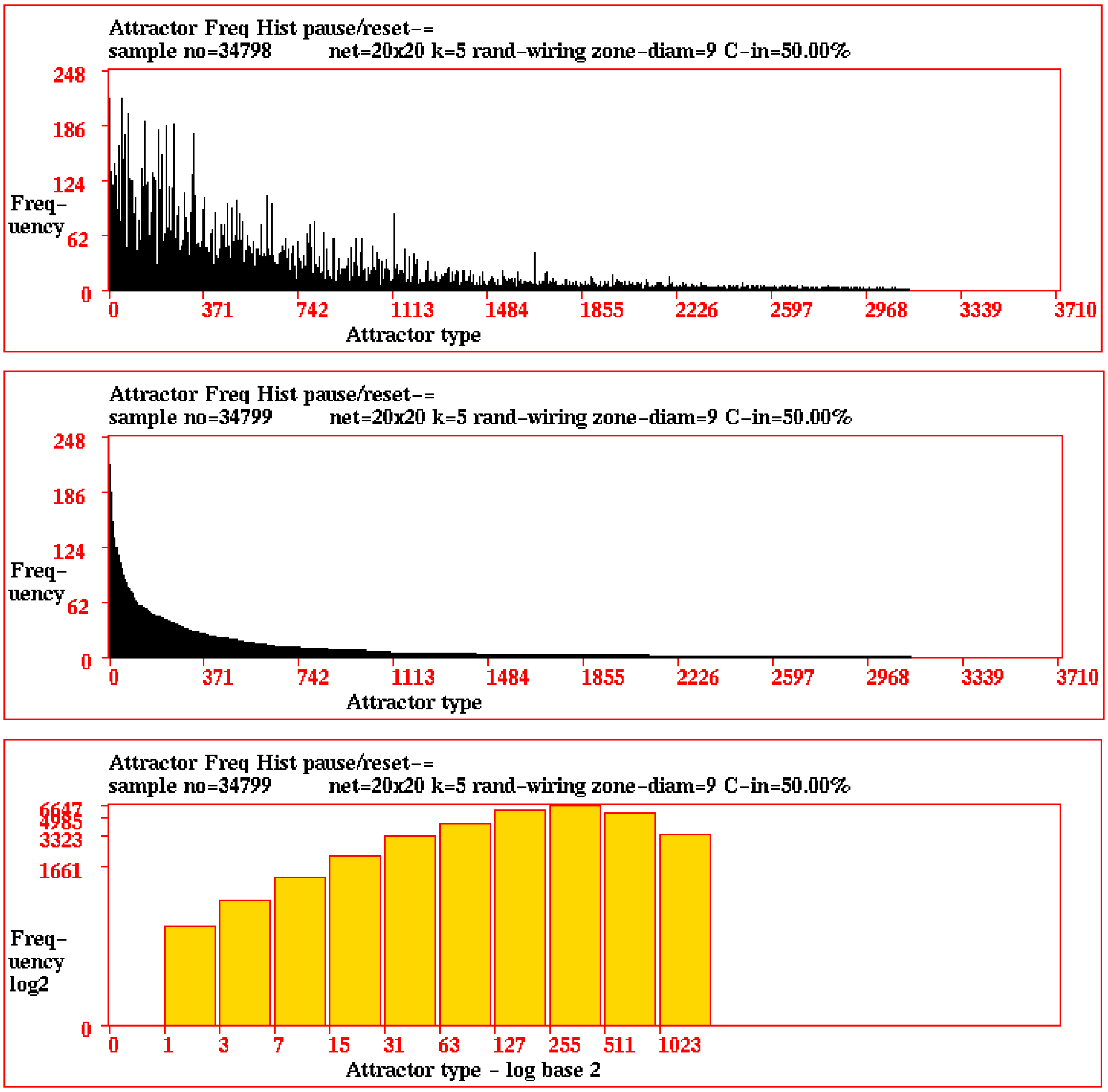


Figure 31.16: The attractor frequency histogram, showing the automatic process for gathering statistics on attractor types. The histogram was paused at sample 34799. This example is for a 20×20 RBN, k=5, with canalizing inputs set to 50%, (see chapter 15). At each pass, a new random initial state is assigned to the network, and the attractor type it falls into is identified. If the type is new it will be added as a new type. The histogram is continuously updated, showing the attractor type (x axis) against the frequency of arriving at that type (y axis), which indicates the relative size of the basin of attraction. *top:* The unsorted histogram. *center:* The sorted histogram, according to frequency. *bottom:* The sorted histogram, shown as a log-log plot.

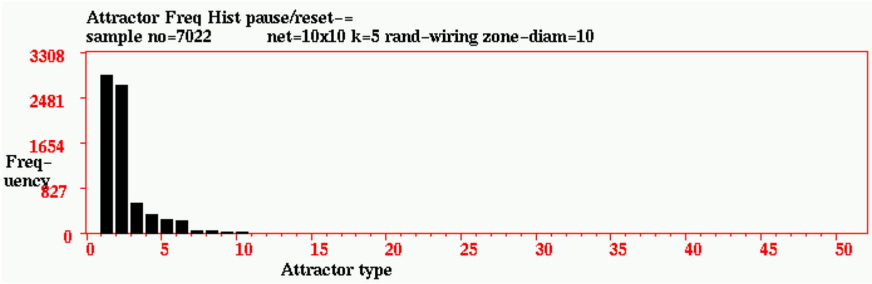
31.18.1 Pausing the attractor histogram

While the histogram is being drawn, enter = (the equals sign) to pause. The following top right prompt is presented for the attractor meta-graph (see chapter 20.3), to show data, to sort the order, clear the current histogram and restart the sample, or to redraw the histogram (in case it was covered by other graphics) and to give further options to rescale, save and load.

attractors: data-d sort-s clear-c redraw/options-o cont-ret::

The attractor meta-graph, showing data and sorting are described below.

31.18.2 Attractor histogram meta-graph



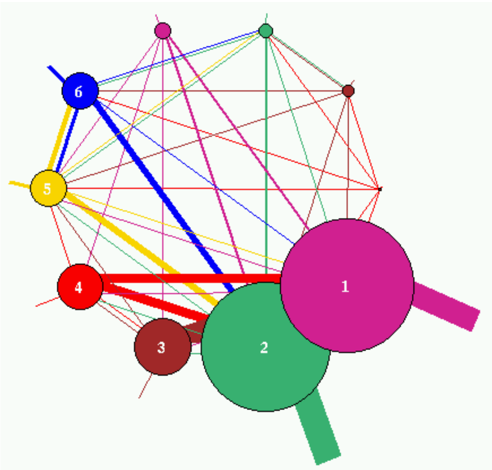


Figure 31.17: *above*: The attractor frequency histogram for an RBN with just Altenberg rules (see section 16.5), k=5, n=100. 10 basins have been found after a sample of about 7000 initial states. *left*: The meta-graph of the attractor histogram (see chapter 20.3), Nodes are scaled to reflect basin volume.

The attractor metagraph, though mainly applied when drawing the complete basin of attraction field (see chapter 20.3), can also be computed for the attractor histogram, though this may take a long time for a histogram with many long period attractors.

Enter **m** in section 31.18.1 to show the meta-graph of the attractor histogram. While being computed the following top right message is shown (enter **q** to abandon),

computing meta-graph, quit-q, or wait...

and the progress is displayed in the xterm window (not for DOS). The basin number is shown progressively as each basin is computed, for example,

```
basins=10 max_period=6 max_trans=53
 1 2 3 4 5 6 7 8 9 10
```

31.18.3 Attractor histogram data

Enter **d** in section 31.18.1 to show the data. A list of attractor types, as many as will fit, will be displayed in a window on the right of the screen, followed by further options.

The column headings denote the following,

type ... the attractor type index.
no ... the number of this attractor type found.
freq ... the frequency of this type
period ... the attractor period.
avtrans ... the average transient (or run-in) length.

The list order depends on whether the list was sorted in section 31.18.4, if not, the list is ranked in the order that attractor types were found.

This example shows the data for the network in figure 31.16 (center), sorted according to attractor frequency,

attractor types so far=3135
type no freq period avtrans
1 217 0.0062 12 29.4
2 216 0.0062 12 29.5
3 200 0.0057 12 28.3
... *(the list continues)*
39 113 0.0032 12 23.1
40 110 0.0032 4 28.6
41 107 0.0031 12 24.0
quit-q jump-j: *(if the complete list fits the window, as in this case)*
or
more-ret quit-q jump-j : *(if more than one window is required, as in this case)*
or
cont-ret jump-j : *(if the end of the list is visible)*

Enter **q** to quit the list and continue the histogram, **j** to jump to a new type index, and **return** to see more of a long list, or to quit the list and continue if the end of the list is visible.

31.18.3.1 Histogram data for density rules

Another example shows the data for a CA, n=149, k=7, rule (hex) fa f3 ca fo fa ff 88 e8 fa 73 0a 00 3a 0c 8a 28, one of the "density rules" mentioned above, where the option in section 31.18 was set to include data on initial states that fall into the all 0s or all 1s point attractors. An additional column, headed **den-/50%/+**, shows the percentage of initial states where the density of 1s was less than 50%, exactly 50%, and more than 50%, and the % error in categorizing the seeds (**e=**), for example,

attractor types so far=2
type no freq period den-/50%/+ avtrans
1 1121 0.4706 1(all 0s) 985/0/136 e=12% 84.2
2 1261 0.5294 1(all 1s) 212/0/1049 e=16% 77.8
cont-ret jump-j :

31.18.4 Sorting the attractor histogram

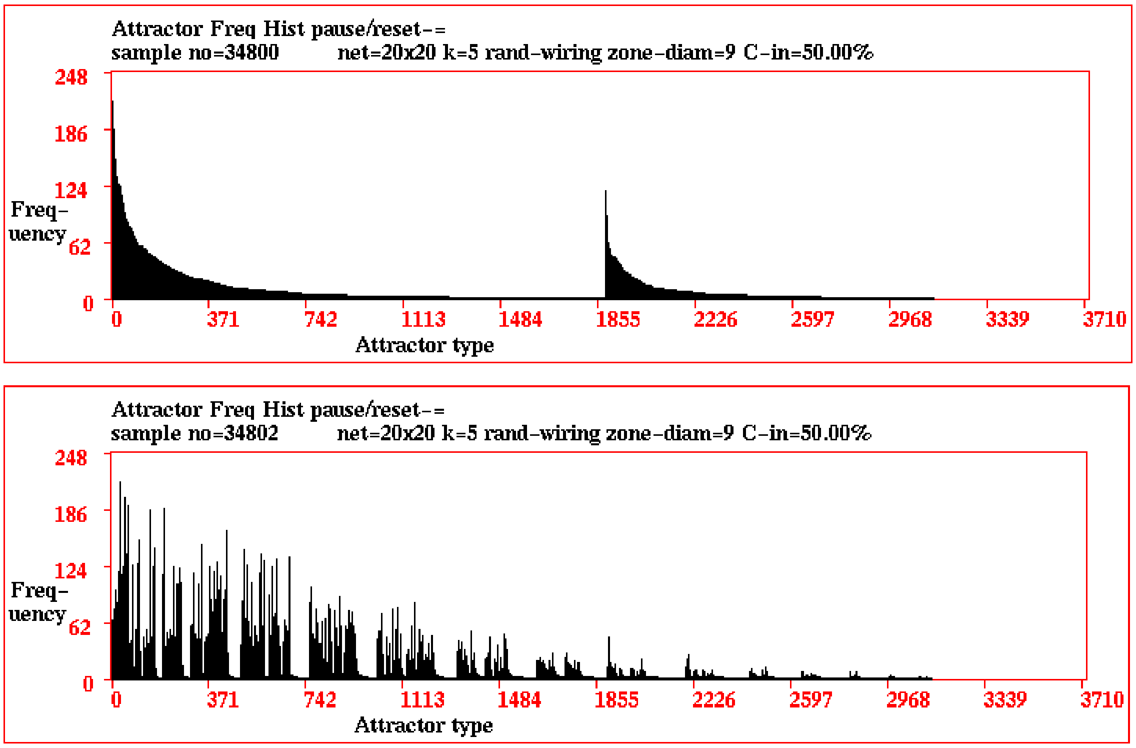


Figure 31.18: The attractor frequency histogram was first sorted according to frequency (figure 31.16, center). It was then by sorted by sorted by period (*top*), then by average transient (*bottom*.

Enter **s** in section 31.18.1 to sort the histogram. The following top right prompt is presented,

sort by: attperiod-a avtrans-t (default freq):

Initially the attractor types are ranked in the order they were found, as in figure 31.16 (top). Enter **return** to sort by frequency. An example is shown in figure 31.16 (center). This corresponds to sorting according to the size of the basin of attraction, as the probability of falling into a basin depends on its size.

Enter **a** to sort by attractor period, **t** to sort by average transient (run-in) length. The result of the new sorting depends on how the list is currently sorted. Having first sorted by frequency, the examples in figure 31.18 show the data sorted first by the attractor period, then by the average transient.

31.18.5 Rescaling the attractor histogram

If **s** to sort, or **o** to redraw the histogram and provide further options, is entered in section 31.18.1, a top right window is presented with options as follows,

undo pause-u save/load-l/s next-ret
redraw-r x-axis-x logx-X logy-y log-both-b:

These options are explained below,

undo pause-u ... enter **u** to continue the histogram without pause.
save/load-l/s ... enter **s** to save the historam data as a `.rul` file (see Filing, chapter 35). Enter **l** to load the data and see it in the Xterm window (not for DOS). The file holds frequencies (0-max) in successive pairs of bytes.
next-ret ... enter **return** to continue the histogram, but pause after the next sample.
redraw-r ... enter **r** to redraw the histogram, which may have been overwritten by other graphics.
x-axis-x ... enter **x** to rescale the x-axis, the following top right prompt is presented,

> **revise cut-off damage size, now 10000 (min 20, def 10000):**

enter the new cut-off. Any damage above the cut-off is lumped together in one excess bin.
logx-X ... enter **x** to show the x-axis in log form, with the following top right reminder,

> **x-axis bined y powers of 2, cont-ret:**

logy-y ... enter **y** to show the y-axis in log form, with the following top right reminder,

> **y axis log2, cont-ret:**

log-both-b ... enter **b** to show both the x and y-axis in log form (as in figure 31.16, bottom), with the following top right reminder,

> **y axis log2, x-axis bined y powers of 2, cont-ret:**

31.18.6 Attractor histogram window

While the attractor histogram is being drawn, information is displayed at the top of the histogram window, including some reminders about the network (see figure 31.14). This information is decoded as follows,

pause/reset-= ... reminder: enter = to pause and reset parameters.
sample no= ... the size of the sample so far.
net=20×20 ... network size (for example).
k= ... neighbourhood size, k.
k=3-5... for a range of k (for example).
local wiring ... local wiring, as in CA
rand-wiring ... random wiring, as in RBN
zone diam=... the size of the relative local zone within which random wiring is confined, see section 11.5.2.
C-in= ... the percentage of canalizing inputs.

To pause, enter = (the equals sign) for the histogram options described in sections 31.18.1 to 31.18.5. Alternatively, pause by entering **q** for the general pause options for space-time patterns described in section 32.13.

The space-time pattern on-the-fly options work during the histogram routine. To speed the attractor histogram, toggle the space-time graphics off with the on-the-fly option **S** (see section 32.8.1.

31.19 Skeleton histogram

Enter **s** at the prompt in section 31.17 above to generate the frozen skeleton histogram.

In this context network elements are refered to as "genes" as the method is applied mainly to study models of genetic regulatory networks.

Five parameters need to be set to define a skeleton type, as follows,

1. **1s only** The frozen pattern under consideration, 0s only, 1s only, or both 0s and 1s.
2. **g** The number of time-steps that a gene must remain unchanged to be considered frozen.
3. **s** The number of time-steps the *pattern* of frozen genes must remain unchanged (within a set Hamming distance) to be considered a frozen skeleton type.
4. **sp** The skeleton percentage Hamming distance limits in (3) above.
5. **tp** How close (within a percentage Hamming distance) must a skeleton type be to a pre-established type to qualify as that type.

A series of five top right prompts are presented to set these parameters,

frozen skeletons: 0s-0 1s-1 both-(def):
no of steps frozen: gene (now 20): skeleton (now 20):
% frozen (def-100): % same type (def-100):

When the histogram is being drawn (see section 31.19.5, a representation of the skeleton is shown in the top center of the screen, noting if the skeleton consists of frozen 0s, 1s, or both.

At the same time, a top right data window gives current data, the types of skeletons found, the current type, and average transient length, for example,

types=639 this=25 avtrans=53.2

31.19.1 Pausing the skeleton histogram

While the histogram is being drawn, enter = (the equals sign) to pause.

The following top right prompt is presented to show data, to sort the order, clear the current histogram and restart the sample, or to redraw the histogram, in case it was covered by other graphics.

skeletons: data-d sort-s clear-c redraw/options-o cont-ret:

Showing data and sorting are described below.

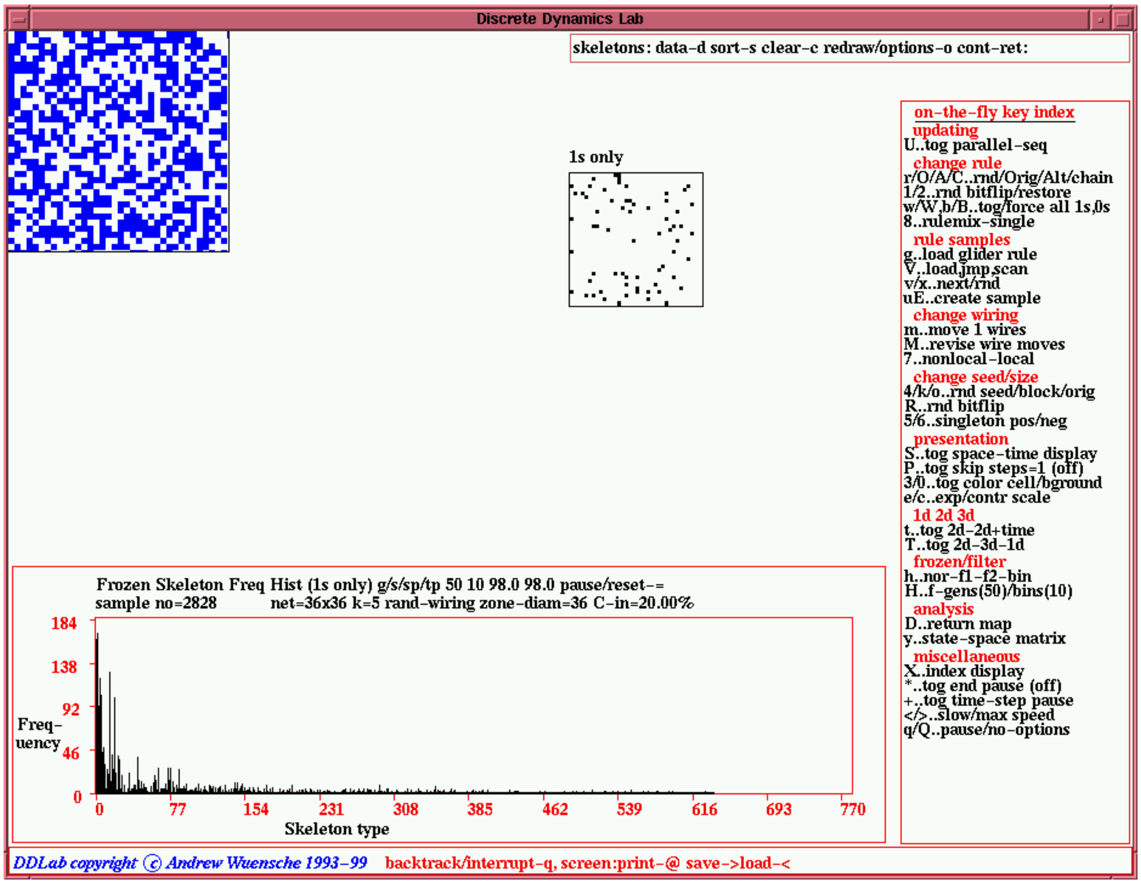


Figure 31.19: The DDLab screen, showing the automatic proceess for gathering statistics on frozen skeleton types. The histogram was paused at sample 2834. This example is for a 36×36 RBN, k=5, with canalizing inputs set to 20%, see chapter 15). The parameters in section 31.19 (for frozen 1s only), labelled **g/s/sp/tp** are 50, 10 98% 98%. At each pass, a new random initial state is assigned to the network, and the skeleton type it falls into is identified. If the skeleton cannot be matched with a pre-existing skeleton type it is added as a new type. The histogram is continuously updated, showing the skeleton type (x axis) against the frequency of arriving at that type (y axis), which indicates the relative size of the quasi basin of attraction. *top left:* A snapshot of the space-time pattern. *top centre:* The frozen skelaton, pattern of frozen 1s. *bottom:* The unsorted histogram.

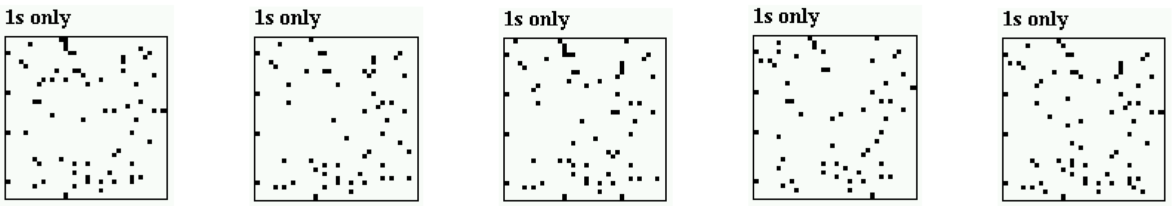


Figure 31.20: Examples of diffeent skeletons, consisting of frozen 1s, from the histogram in figure 31.19. The current skeleton is shown in the top centre of the screen as the histogram is updated.

31.19.2 Skeleton histogram data

Enter **d** in section 31.19.1 to show the data. A list of skeleton types, as many as will fit, will be displayed in a window on the right of the screen, followed by further options.

The column headings denote the following,

type ... the skeleton type index.
no ... the number of this skeleton type found.
freq ... the frequency of this type
%fr-0s-1s ... the percentage of frozen 0s and 1s making up the skeleton.
avtrans ... the average transient (or run-in) length.

The list order depends on whether the list was sorted in section 31.19.3, if not, the list is ranked in the order that attractor types were found.

This example shows the data for the network in figure 31.16 (center), sorted according to attractor frequency,

skeleton types so far=3135
type no freq %fr-0s-1s avtrans
1 165 0.0581 5.3 4.6 53.5
2 171 0.0602 5.5 5.6 53.3
3 94 0.0331 5.4 5.2 53.5
... *(the list continues)*
39 9 0.0032 5.6 5.7 52.8
40 9 0.0032 6.0 5.0 52.6
41 9 0.0032 5.6 5.2 55.1
quit-q jump-j: *(if the complete list fits the window, as in this case)*
or
more-ret quit-q jump-j : *(if more than one window is required, as in this case)*
or
cont-ret jump-j : *(if the end of the list is visible)*

Enter **q** to quit the list and continue the histogram, **j** to jump to a new type index, and **return** to see more of a long list, or to quit the list and continue if the end of the list is visible.

31.19.3 Sorting the skeleton histogram

Enter **s** in section 31.19.1 to sort the histogram. The following top right prompt is presented,

sort by: frozen-0-1-b avtrans-t (default freq):

Initialy the skeleton types are ranked in the order they were found, as in figure 31.19. Enter **return** to sort by frequency. as in in figure 31.21. This corresponds to sorting according to the fraction of state-space "drained" by the skeleton.

Enter **t** to sort by average transient (run-in) length.

The histogram can also be sorted by number of frozen 0s, 1s or both 0s and 1s, enter **0**, **1** or **b**.

The result of the new sorting depends on the on how the list is currently sorted. Having first sorted by frequency, the examples in figure 31.21 show the data sorted first by the attractor period, then by the average transient.

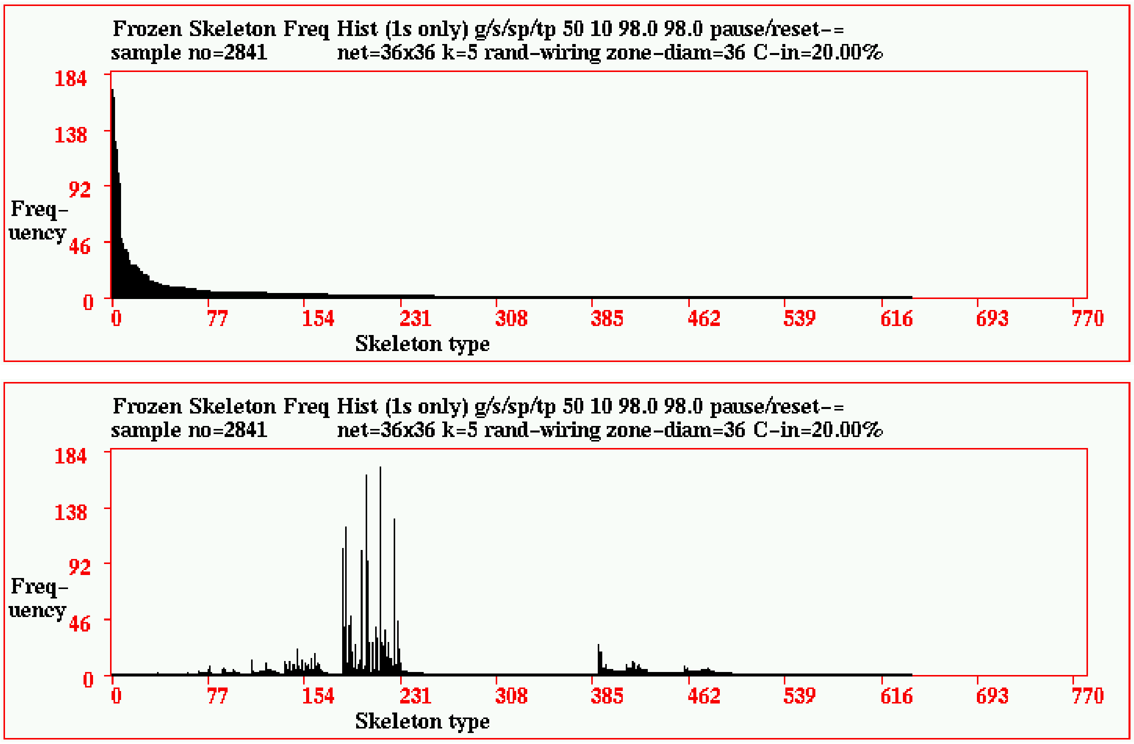


Figure 31.21: The attractor frequency histogram was first sorted acording to frequency (*top*), then by average transient (*bottom*.

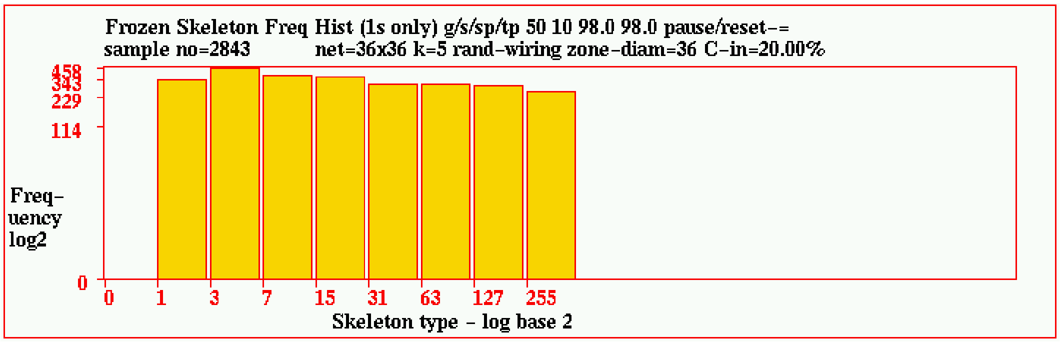


Figure 31.22: The skeleton frequency histogram shown as a log-log plot. The histogram in figure 31.19, was first sorted by frequency (figure 31.21), then rescaled as a log-log plot.

31.19.4 Rescaling the skeleton histogram

If **s** to sort, or **o** to redraw the histogram and provide further options, is entered in section 31.19.1, a top right window is presented with options as follows,

> **undo pause-u save/load-l/s next-ret**
> **redraw-r x-axis-x logx-X logy-y log-both-b:**

These options are the same as for the attractor histogram and are explained in section 31.18.5.

31.19.5 Skeleton histogram window

While the skeleton histogram is being drawn, information is displayed at the top of the histogram window, including some reminders about the network (see figure 31.14). This information is decoded as follows,

(1s only) ...	
(0s only) ...	
(0s or 1s) ...	reminder of the basis of the skeleton.
g/s/sp/tp ...	see below, and section 31.19.
g ...	gene frozen time-steps.
s ...	pattern frozen time-steps.
sp ...	skeletonfrozen time-steps.
s ...	skeleton % Hamming distance.
tp ...	skeleton type % Hamming distance.
pause/reset-= ...	reminder: enter = to pause and reset parameters.
sample no= ...	the size of the sample so far.
net=36×36 ...	network size (for example).
k= ...	neighbourhood size, k.
k=3-5...	for a range of k (for example).
local wiring ...	local wiring, as in CA
rand-wiring ...	random wiring, as in RBN
zone diam=...	the size of the relative local zone within which random wiring is confined, see section 11.5.2.
C-in= ...	the percentage of canalizing inputs.

To pause, enter = (the equals sign) for the histogram options described in sections 31.19.1 to 31.19.4. Alternatively, pause by entering **q** for the general pause options for space-time patterns described in section 32.13.

The space-time pattern on-the-fly options work during the histogram routine. To speed the skeleton histogram, toggle the space-time graphics off with the on-the-fly option **S** (see section 32.8.1.

Chapter 32

Drawing space-time patterns, and changes on-the-fly

Space-time patterns are drawn if **s** was selected at the first output parameter prompt (section 24.1), followed by the various options in chapter 31, "Output parameters for space-time patterns" (enter **d** to accept all defaults). Alternativly, **s** can be entered at the final attractor options for single basins (sections 29.3 and 29.3.1). Note that a "single basin/subtree" would have been selected at the first DDLab prompt in section 6.1 by entering **s**.

This chapter describes parameters and options that can be reset or activated while space-time patterns are being drawn (on the left of the screen), either on-the-fly, or by further prompts if the drawing is interupted.

On-the-fly options include: changes to updating, rule/s, wiring, seed and presentation, showing "frozen" regions and filtering, and various methods of analysis. There are also options for loading glider rules, and automaticaly classifying rule-space (described in greater detail in chapter 33).

If the drawing is interupted, extra options are presented in a top right window. One of these is the network graph which can be rearanged in the many ways described in chapter 20, after which an additional version of the space-time graphics will be run according to the graph layout and node sizes, with corresponding presentation flexibility.

Further options include filing, canalyzing, and further methods for manipulating the space-time patterns. Some of these parameters were first set in section 31, so can be reset without the need to backtrack. Others are new parameteres and options which can only be activated once space-time patterns have started.

32.1 On-the-fly key index

An "on-the-fly key index" window is usually displayed on the right of the screen while space-time patterns are drawn as a reminder of the various options which may be activated on-the-fly from the keyboard. An example is shown in figure 32. The options are divided into categories, with subheadings in red, and may vary for different types of network and current settings. The on-the-fly window can be toggled on-off by entering with **X**. The on-the-fly options are summarized in section 32.2 and explained in greater detail in sections 32.3 - 32.12.6.

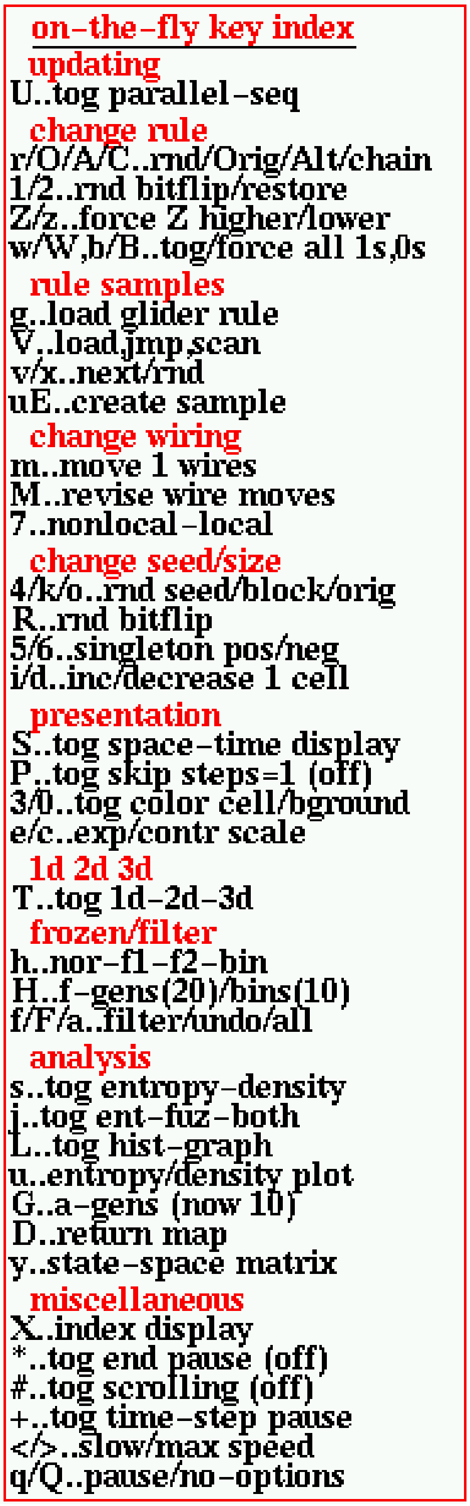


Figure 32.1: An example of the "on-the-fly key index" for space-time patterns (for a 1d CA) which is usually displayed on the right of the screen while space-time patterns are drawn on the left side, as a reminder of the various on-the-fly options which may be activated from the keyboard.
The options are divided into subcategories, with subheadings in red. They are context dependent, and may vary for different types of network or according to current settings.
The window can be toggled on-off with the **X** key.

32.2 Summary of on-the-fly options for space-time patterns

updating

U..parallel-seq . . . toggle parallel/sequential updating (section 32.3)

change rule

r/O/A/C..rnd/Orig/Alt/Chain . . . new rule/s (section 32.4.1)

r . . . random rule
O . . . restore original rule
A . . . Altenberg rule
C . . . Chain rule

1/2..rnd bitflip/restore . . . mutate/restore rule/s (section 32.4.2)

1 . . . mutate rule-table by 1 random bit
2 . . . restore mutated bits in reverse order

Z/z..force Z higher/lower . . . change rule's Z-parameter (section 32.4.3)

Z . . . force Z higher
z . . . force Z lower

w/W,b/B..tog/force all 1s,0s . . . change the "hot bits" (section 32.4.4)

w . . . flip the output of all 1s
W . . . force the output of all 1s to 1
b . . . flip the output of all 0s
B . . . force the output of all 0s to 0

8..rulemix-single . . . toggle rule-mix/single rule at cell 0 (section 32.4.5)

rule samples

g..load glider rule . . . load a glider rule (section 32.5.1)
V..load,jmp,scan . . . load rule sample (section 32.5.2 and chapter 33)
v/x..next/rnd . . . scan successive rules
x . . . jump to a random sample index
uE..create sample . . . create sample of classified rule-space (section 32.5.4 and chapter 33.

change wiring

m..move 1 wires . . . randomly move one or more wires (section 32.6)
M....revise wire moves . . . change number of wires to move (section 32.6.2)
7..nonlocal-local . . . toggle non-local/local wiring (section 32.6.3)

change seed/size

4/k/o..rnd seed/block/orig . . . change state/seed (section 32.7.1)

4 . . . random state
k . . . random block

o ...	restore original state
R..rnd bitflip ...	mutate state by 1 random bit (section 32.7.2)
5/6..singleton pos/neg ...	mutate state by 1 random bit (section 32.7.3)
5 ...	set positive singleton seed
6 ...	set negative singleton seed
i/d..inc/decrease 1 cell ...	change network size (section 32.7.4)
i ...	increase network size by 1 cell
d ...	decrease network size by 1 cell
presentation	
S..tog space-time display ...	toggle the display of space-time graphics on/off (section 32.8.1)
P..tog skip steps=1 (off) ...	toggle skipping time-steps (section 32.8.2)
3/0..tog color cell/bground ...	toggle skipping time-steps (section 32.8.3)
3 ...	toggle cell colors neighborhood/value
0 ...	toggle background color white/light green
$..tog sound ...	toggle sound *DOS only* (section 32.8.4)
e/c..expand/contr scale ...	expand/contract cell scale (section 32.8.5)
e ...	expand cell scale by 1 pixel
c ...	contract cell scale by 1 pixel
1d 2d 3d	change dimension (section 32.9)
T..tog 1d-2d-3d ...	toggle 1d/2d/3d (section 32.9.1)
t..tog 2d-2d+time ...	toggle 2d/2d+time (section 32.9.2)
p..2d draw plane ...	draw 2d plane in 2d+time (section 32.9.3)
I..tog 3d slice ...	toggle 3d in successive slices (section 32.9.4)
frozen/filter	(section 32.10)
h..nor-f1-f2-bin ...	toggle between different ways of showing "frozen" cells (section 32.10.1)
H..f-gens (now 20)/bins(10) ...	change "frozen" parameters (section 32.10.2)
f/F/a..filter/undo/all ...	(section 32.10.5)
f ...	progressively filter
F ...	progressively unfilter in reverse order
a ...	restore all filtering
analysis	
s..tog entropy-density ...	toggle input-entropy/pattern density (section 32.11.1)
j..tog ent-fuz-both ...	toggle between different input-entropy presentions (section 32.11.2)

L..tog hist-graph ...	show the input-frequency histogram with a time dimension (section 32.11.3)
u..tog entropy/density graph ...	show the density/entropy scatter plot (section 32.11.4)
G..a-gens (now 10) ...	change analysis generations (section 32.11.5)
D..return map ...	show/delete return map (section 32.11.6)
y..state-space matrix ...	show/delete state-space matrix (section 32.11.7)
miscellaneous	
X..index display ...	show/delete the on-the-fly key index (section 32.12.1)
***..tog end pause (on)** ...	toggle the end pause on/off (section 32.12.2)
#..tog scrolling ...	toggle the scrolling on/off (section 32.12.3)
+..time-step pause ...	pause after each time-step (section 32.12.4)
</>..slow/max speed: ...	speed (section 32.12.5)
< ...	slow down the space-time patterns
> ...	restore full speed
q/Q..pause/no-options ...	(section 32.12.6)
q ...	pause space-time patterns and display data/options
Q ...	turn off data/options when pausing

32.3 updating

U..parallel-seq: enter **U** to toggle on-the-fly between parallel and sequential updating (see also section 31.2). The current status is given by the prompt order, i.e. **U..seq-parallel** indicates that sequential updating is current.

32.4 change rules

These on-the-fly options amend (mutate) the rule, or the rules in a mixed rule network.

32.4.1 rnd/Orig/Alt/Chain

r/O/A/C..rnd/Orig/Alt/Chain: enter **r**, **O**, **A** or **C** to amend rule/s as described below,

r ... enter **r** to assign a new random rule. For mixed rule networks, random rules are assigned for all cells in the network. For a single rule network, the Z-parameter and the rule (for $\boldsymbol{k} \leq \mathbf{5}$) is shown in a top right window, for example,

Z=0.671875 hex=bc a9 15 4b

O ... enter **O** to restore the rule that was originaly selected. For mixed rule networks the rule is assigned to cell index 0 only.

A . . . enter **A** set an an "Altenberg" rule, chosen at random (see section 16.5). For mixed rule networks, Altenberg rules are assigned for all cells in the network. For a single rule network, the Z-parameter and the rule (for $\boldsymbol{k} \leq \mathbf{5}$) is shown in a top right window, for example,

Z=0.507812 hex=eb 48 e8 c8

C . . . enter **C** set a maximally chaotic rule, or "Chain" rule, chosen at random (see section 16.7). For mixed rule networks, chain rules are assigned for all cells in the network. For a single rule network, the $\boldsymbol{Z_{left}}$ and $\boldsymbol{Z_{right}}$ parameters, and the rule (for $\boldsymbol{k} \leq \mathbf{5}$), are shown in a top right window, for example,

zl=0.875 zr=1 hex=97 96 68 69

32.4.2 rnd bitflip/restore

1/2..rnd bitflip/restore: enter **1** or **2** to mutate or restore the rule/s, as described below,

1 . . . enter **1** to mutate the rule by 1 bit, a random bitflip in the rule-table, or a different random bitflip for each network element in a rule-mix. For single rule networks only, data will appear in a top right window, for example,

bitflip=at pos 30, flip=2 Z=0.671875 hex=4e 68 63 46

This shows the cell index of the bitflip, the number of flips so far, the new Z parameter, and the new rule in decimal (for $\boldsymbol{k} \leq \mathbf{3}$) and hex (for $\boldsymbol{k} \leq \mathbf{5}$).

2 . . . for single rule networks only, enter **2** to restore the random bit-flips above, or the sequence of bitflips in reverse order. Data will appear in a top right window, for example,

bit back =at pos 24, flip=0 Z=0.671875 hex=0f 68 63 46

This shows the cell index of the bitflip, the flip backs in reverse order (0 means the original rule has been restored), Z parameter, and the rule in decimal (for $\boldsymbol{k} \leq \mathbf{3}$) and in hex (for $\boldsymbol{k} \leq \mathbf{5}$).

32.4.3 force Z higher/lower

for single rule networks only

Z/z..force Z higher/lower: enter **Z** (upper case) to progressively force the Z-parameter higher, or enter **z** (lower case) to force it lower, as in section 16.9.3. For a single rule network, the Z-parameter and the rule (for $\boldsymbol{k} \leq \mathbf{5}$) is shown in a top right window, for example,

Z=0.53125 hex=77 6f f1 8f

Another window below shows more information: canalizing, $\boldsymbol{Z_{left}}$, $\boldsymbol{Z_{right}}$, $\boldsymbol{\lambda}$-parameter, $\boldsymbol{\lambda_{ratio}}$, $\boldsymbol{P}$-parameter, and Z-parameter, for example,

C=0/5 zl=0.375 zr=0.53125
ld=0.6875 ld-r=0.625 P=0.6875 Z=0.53125

32.4.4 tog/force all 1s, 0s

w/W,b/B..tog/force all 1s,0s: these options affect the "hot bits" at the extreme ends of the rule-table, for the neighbourhoods all 1s and all 0s.

w/W ... enter **w** to flip the output of the all 1s neighborhood, or enter **W** to force the all 1s neighborhood to output a 1. This applies for a single rule, or for all rules in a mixed rule network.

b/B ... enter **b** to flip the output of the all 0s neighborhood, or enter **B** to force the all 0s neighborhood to output a 0. This applies for a single rule, or for all rules in a mixed rule network.

32.4.5 rulemix-single

for a mixed rule network only

8..rulemix-single: enter **8** to toggle between running the network according to the its rule-mix, or according to the single rule at cell index 0. The current status is given by the prompt order, i.e. **8..single-rulemix** indicates that the single rule is current.

32.5 rule samples

These on-the-fly options relate to loading "glider rules", and to automatically classifying rule-space[19], including creating, sorting, analysing and probing the sample. The latter is a larger topic described in detail in chapter 33.

32.5.1 load glider rule

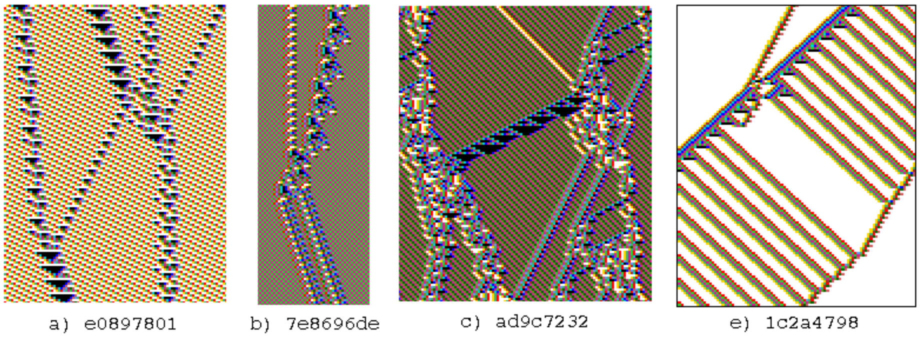


Figure 32.2: Examples of k=5 complex space-time patterns with interacting gliders from the sample of glider rules included with DDLab, which can be loaded on-th-fly while space-time patterns are being drawn. The $\boldsymbol{k}$=5 rule numbers are shown in hex.

g..load glider rule: enter **g** to load a "glider" rule from a file. A collection of "complex" rules that produce coherent interacting structures (known as gliders) are provided in separate files with

DDLab. The samples currently available are for k=5, 6 and 7 rules only, for 1d networks, though the command will also work for 2d and 3d. These are held in the files `glider5.r_s` (62 rules), `glider6.r_s` (31 rules) and `glider7.r_s` (31 rules). The glider rule will become the new rule in a single rule network. For a mixed rule or mixed k network, provided that k for cell index 0 is 5, 6 or 7, **g** will load a glider rule at cell index 0. This rule can be applied to the whole network by toggling **8..rulemix-single** in section 32.4.5 above. The new glider rule will be shown in a top right window. For example (for k=6)

1d glider rule=2 hex=b3 87 b6 b8 8c d4 af a0

rule=2 indicates that the rule is number 2 in the current sample.

32.5.2 load,jmp,scan

for single rule networks only

V..load,jmp,scan: enter **V** to load, display, scan and probe a sample of automatically classified rule-space. Various top right prompts will appear, firstly to allow loading the file (see Filing, chapter 35). The samples currently provided with DDLab are for 1d k=5, 6 and 7 rules only, and are held in the files `five5ss.sta`, `six5ss.sta` and `sev5ss.sta`. New samples can be created (see chapter 33). Once loaded, a further top right prompt is presented as follows,

list: all-l pos+p, sort-s, quit-q
entropy sandard dev plot: Z-Z lda-L ent-def, smalldots+d:

Enter **q** to skip the further options and continue with the space-time pattern. Otherwise enter **return** to display the mean entropy - standard deviation scatter plot with various further options, for example to probe points on the plot to select rules, and to display the scatter plot in 3d with a vertical frequency axis. These further options are explaned in chapter.

Once a rule sample has been loaded, enterering **V** allows loading a new rule sample, or scanning the space-time patterns in the current sample. The following top right prompt is presented, for example if the k=5 file `five5ss5.sta` was loaded, which contains 17680 rule,

rule sample: load new-n, rule index/scan 1-17680: *(values shown are examples)*

Enter a number in the range to jump to a sample index. The following further prompr is presented to allow scanning space-time patterns in blocks, for example if index 72 was entered above,

scan in blocks from index 72 -i: *(values shown are examples)*

Enter **return** to select the rule at index 72 in the sample and view its space-time patterns in the normal way, or enter **i** to scan successive rules in blocks of time-steps starting at index 72.

If **i** is selected, a further prompt, **set time-steps (def 150):** alows the number of time-steps in the block to be resized.

It is best to toggle "scrolling" off when scanning successive blocks, with on-the-fly option # (see section 32.12.3). Furthermore, the blocks will scan continuously unless the "end pause" option is toggled on with on-the-fly option * (see section 32.12.2). Figure 32.3 shows the DDLab screen scanning in blocks of 88 time-steps. The current rule is shown in hex in a top right window, for exanple,

index 78 mean-entropy=0.0565 standard-deviation=0.1214
rule=dd 88 e8 74

Note that with "end pause" on, the pause prompt described in section 32.14 will appear. This can be suppressed if required by entering **Q** (see section 32.12.6). Enter **return** for the next set of blocks. To exit scanning in block, interupt with **q**.

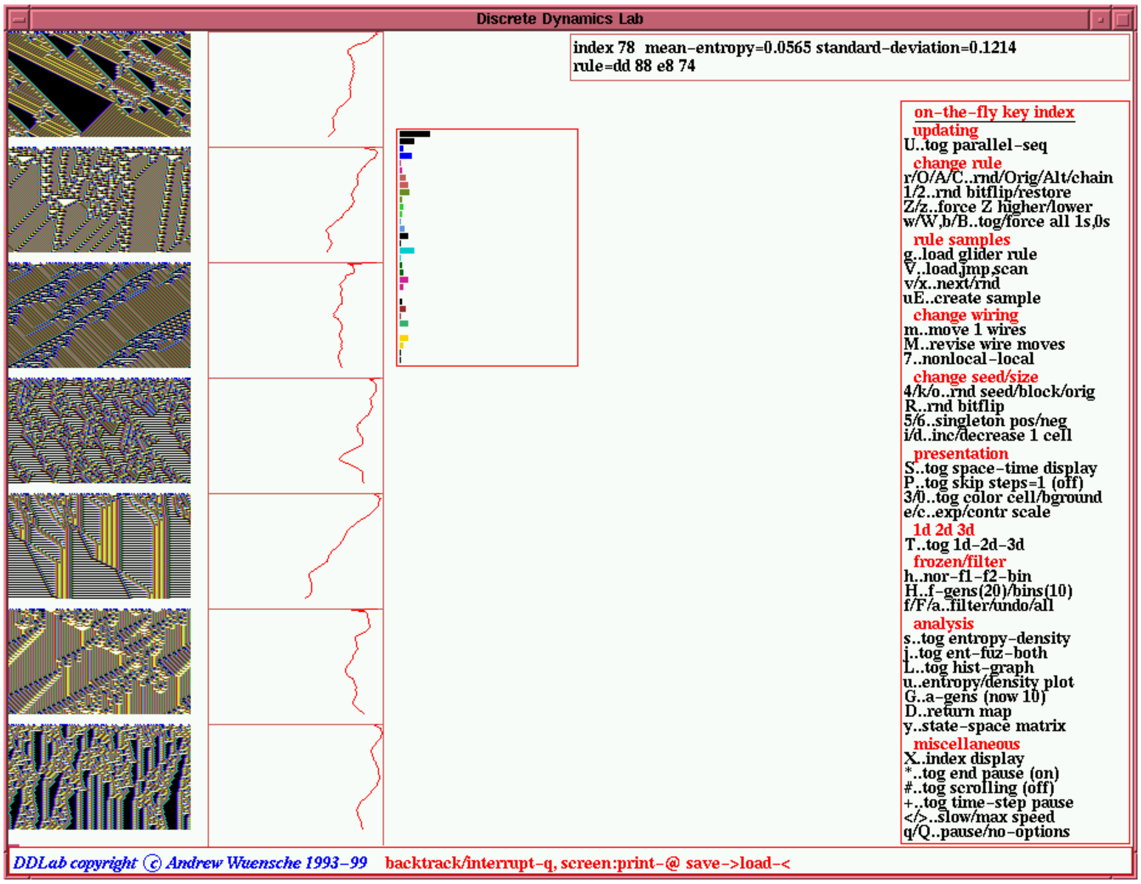


Figure 32.3: Scanning rule-sample space-time patterns in blocks. In this case the k=5 file `five5ss.sta`, provided with DDLab, was first loaded, then scanning from sample index 72. The blocks wee resized to 88 time-steps, n=150.

32.5.3 next/rnd

for single rule networks only

v/x..next/rnd: once the rule-space sample has been loaded (with **V** above, or in section 32.14.2), enter **v** to scan the space-time patterns of rules successive in the sample, or **x** to jump to a random sample index. In both cases a top right window shows the sample index, the rule's mean-entropy and standard-deviation (see chapter 33), and the rule itself in hex, for example (for a k=7 rule),

index 316 mean-entropy=0.1165 standard-deviation=0.0805
rule=ae ae 83 a3 35

Entering **v** will scan successive rules in the sample starting from rule index 0, or from the rule that was "jumped to" with **V** in section 32.5.2.

32.5.4 create sample

uE..create sample: enter **u** followed by **E** to create a sample of automatically classified rule-space, described in chapter 33. Note that the initial **u** initiates an entropy-density plot (see section 32.11.4). If this plot is current, the on-the-fly prompt is **E..create sample**, so enter just **E** to create a sample.

32.6 change wiring

These options allow network wiring to be changed, mutated, on-the-fly.

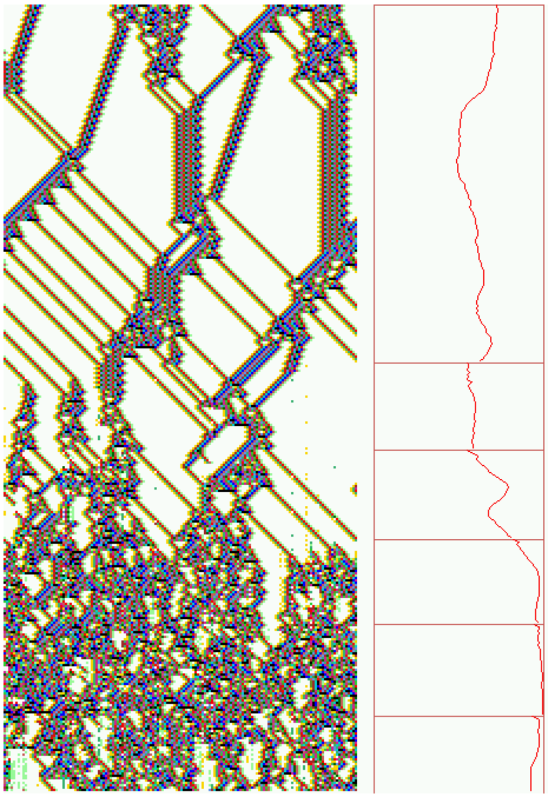


Figure 32.4: Randomizing the wiring of a 1d CA by stages, for rule 6c 1e 53 a8, n=150. The wire moves were first revised to 2% (15 out of 750 wires) with on-the-fly option **M** (section 32.6.2), then the wiring was cumulativly randomized with on-the-fly option **m** (section 32.6.1), at the points indicated by horizontal lines in the input-entropy plot. Note that glider structure is progressively degraded.

32.6.1 move x wires

m..move 1 wires: enter **m** to randomly move one wire or a preset number of wires. For a local 1d, 2d or 3d network, first toggle running the network according to its non-local wiring with key **7** (section 32.6.3) to see the effect of the change.

The number of wires to be moved can be reset with **M** in section 32.6.2 below, and will show up in the on-the-fly prompt, for example,

m..move 25 wires

32.6.2 revise wire moves

M..revise wire moves: enter **M** to change the number of wires to be moved with **m** in section 32.6.1 above. The "mutate wiring" prompts (section 28.3) will be presented in a top right window, for example,

relocal-l, wires to move=1/750=0.133%: all-a number-n percentage-p:

This option also allows the wiring to be made local for 1d, 2d and 3d networks.

32.6.3 nonlocal-local

7..nonlocal-local: enter **7** to toggle between running the network according to its non-local wiring (if it has it), or running as if the wiring was local (for 1d, 2d or 3d networks). The original non-local wiring held in memory remains intact when local wiring is toggled. The current status is given by the prompt order, i.e. **7..local-nonlocal** indicates that the network is currently being run as if it has local wiring. If the wiring held in memory is itself local, there will be no difference between local and nonlocal. However, wire moves made to the local wiring with **m** in section 32.6.1 will only show up if non-local wiring is toggled.

32.7 change seed/size

These on-the-fly options amend, mutate, the current state or seed, and also the of size of the network (for 1d networks 1d only displayed in 1d).

32.7.1 rnd seed/block/orig

4/k/o..rnd seed/block/orig: enter enter **4**, **k** or **0** to amend the seed as described below,

- **4** ... enter **4** to reset the current network state at random, but with the density of 1s bias set in 21.5.3.
- **k** ... enter **k** to reset the network state as a central random block, but with the density of 1s bias of the block set in 21.5.3, (in 1d, 2d or 3d). The size of the block was set in section 21.4, and the density of 1s bias of the block in section 21.5.3.
- **o** ... enter **o** to restore the network state to the original, (as set in section 21).

32.7.2 rnd bitflip

R..rnd bitflip: enter **R** to flip a bit in the current state at a random position.

32.7.3 singleton pos/neg

5/6..singleton pos/neg: enter enter **5** or **6** to reset the seed as a positive or negative singleton seed, a central 1 surounded by 0s, or a central 0 surounded by 1s.

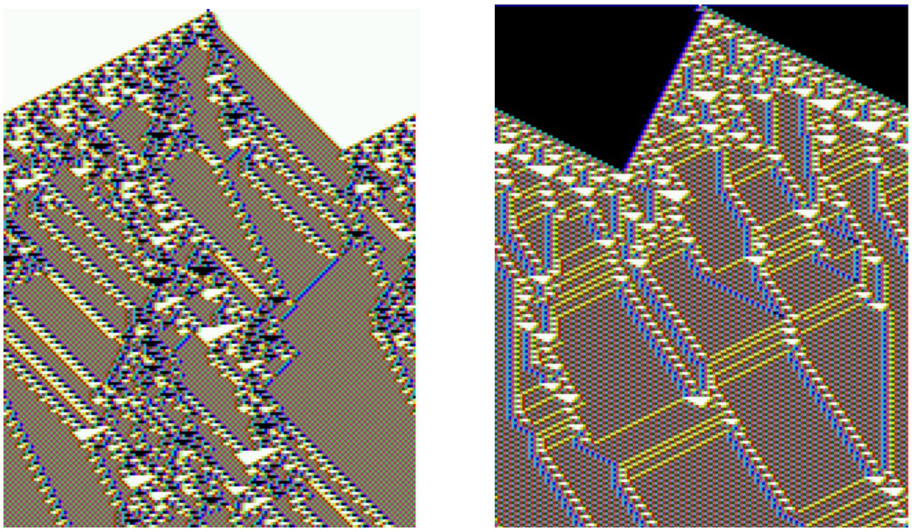


Figure 32.5: *left*: a positive singleton seed, a central 1 surounded by 0s, and *right*: a negative singleton seed, a central 0 surounded by 1s, set on-the-fly by entering **5** or **6**. *left*: rule 96 8c 4f 76, *right*: rule d2 8f 02 2c. n=150, about 190 time-steps.

32.7.4 inc/decrease 1 cell

for 1d networks only, shown in 1d

i/d..inc/decrease 1 cell: enter enter **i** or **d** to increase or decrease the size of the network by 1 cell. The original network is preserved as far as possible. The extra cell is assigned local wiring for networks that were originaly defined as local, otherwise random wiring is assigned. For a mixed rule, homogeneous k network, one of the rules from the current set is assigned to the extra cell. For a mixed k network, a random k in the current range is assigned, and a random rule. The change in size is indicated in a top right window, for exanple,

n increased 166 -> 167 or **n decreased 167 -> 166**.

32.8 presentation

These options amend various aspect of the presentation of space-time patterns on-the-fly, including color and scale.

32.8.1 tog space-time display

S..tog space-time display: enter **S** to toggle the display of space-time graphics on/off. Suppressing space-time patterns, which are still iterating, speeds up other processes, for example creating a sample of automatically classified rule-space (chapter 33), histograms of "damage" (section 31.16), and histograms of attractors or skeletons section 31.17.

32.8.2 tog skip steps

P..tog skip steps=1 (off): enter **P** to toggle between normal space-time patterns and skipping the presentation of a given number of time-steps. The number of time-steps to be skipped can be changed in section 32.14.5. The default is 1, i.e. showing every seconf time-step. The current number of time-steps, and if skipping is currently on or off, is indicated in the prompt, for example,

P..tog skip steps=3 (on)

32.8.3 tog color cell/bground

3/0..tog color cell/bground: These on-the-fly options amend the color of space-time patterns.

3 ... enter **3** to toggle between cell colors according to the neighborhood "looked up" and the actual cell value 0 or 1, represented by a dark and a light color. This can also be preset in section 31.3. Examples are shown in figure 32.10.3 (a,b), and figures 31.1, 31.2 and 31.3 for 1d, 2d and 3d networks.

0 ... if cells are colored by value (see above), and for 1d and 2d networks only, enter **0** to toggle between the default background color, white, and light green (for a white background), or black and dark green (for a white background). this is sometime necessary to show up the background.

32.8.4 tog sound

DOS only

$..tog sound: for DOS only, toggle sound generated by the network. Both the pitch and delay are generated by 8 bits near the center of the network.

32.8.5 expand/contr scale

e/c..expand/contr scale: enter **e** to expand the scale of cells in the space-time pattern by 1 pixel, enter **c** to contract the scale by 1 pixel. This changes the scale of the space-time patern in 1d, 2d or 3d.

32.9 1d 2d 3d

Space-time pattern can be displayed in a dimension other than the "native" dimension. A 1d network can be displayed in 2d or 3d, a 2d in 1d or 3d, and 3d in 1d or 2d. The initial display dimension can be set to someting other than the native dimention in section 31.5, or changes can be toggled on-the-fly, described here. In addition: If a network is being displayed in 2d, it can be toggled between 2d and 2d+time.

If a network is being displayed in 3d, the default display method in Unix and Linux is to show each time-step as one 3d snapshot. This can be toggled to build up successive horizontal slices, similar to 2d+time. For DOS, this is the default and only method.

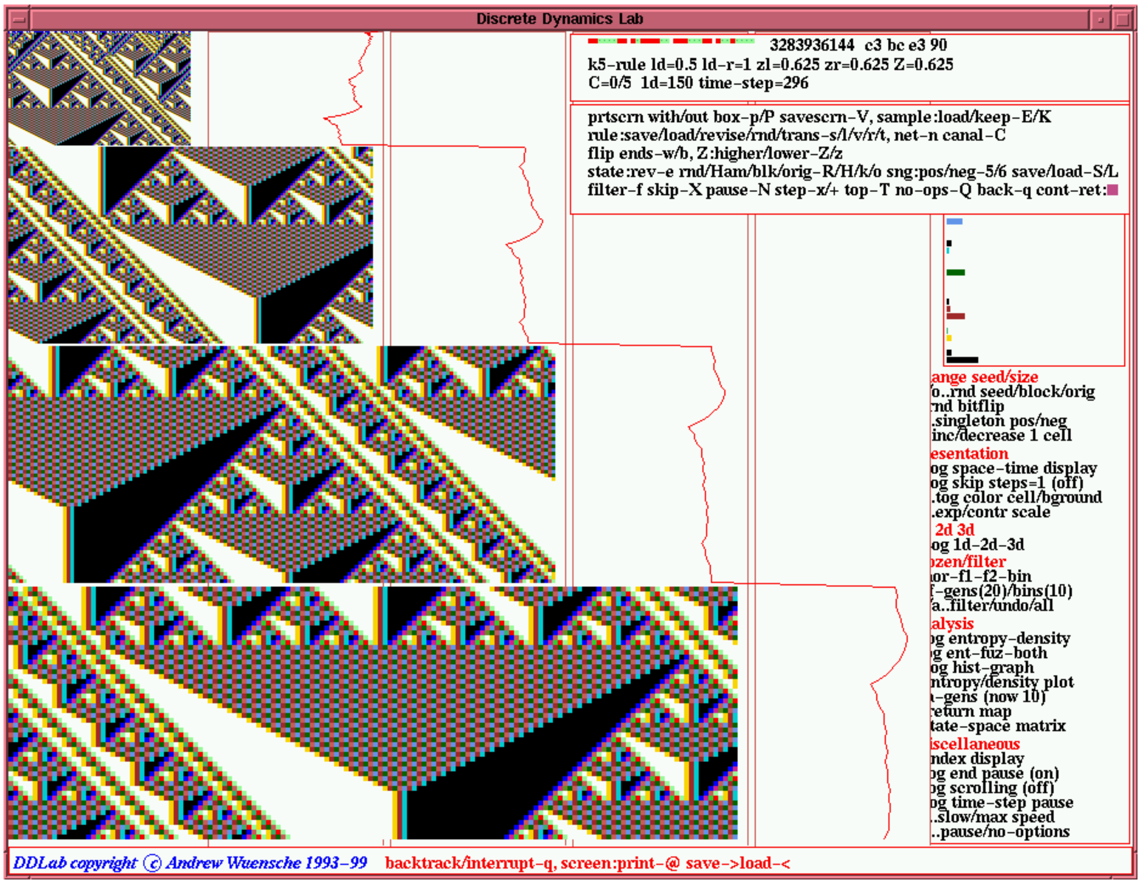


Figure 32.6: Expanding and contracting 1d space-time patterns. At the top the scale of a cell is 1 pixel, then below: 2, 3 and 4 pixels. **e** was entered to incresase the scale by 1 pixel on-the-fly at each key press. **c** contracts the scale by one pixel. n=150, k=5 rule c3 bc e3 90. The network details are shown in the top right window. Below are further prompts described in section 32.14. Expanding and contracting also works for 2d, 2d + time, and 3d networks.

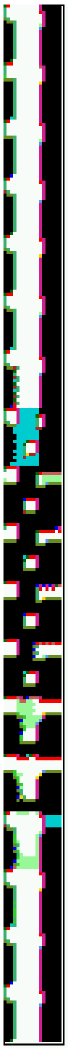
32.9.1 tog 1d-2d-3d

T..1d-2d-3d: if 2d+time is not active (see section 32.9.2 above), **T** is a 3-way toggle to change the display of space-time patterns between 1d, 2d and 3d. Space-time pattern can be displayed in a dimension that is not the "native" dimension as described in section 31.5, from where the presentation dimension can also be pre-selected. The current status is shown in the prompt, i.e. if 2d is active the prompt is **2d-3d-1d**, if 3d is active the prompt is **3d-1d-2d**. Note that only native 1d networks shown in 1d can be scrolled (see section 32.12.3).

The automatic method for converting between dimensions is as follows:

1d to 2d:	$i =$ the smallest factor $\geq \sqrt{\boldsymbol{n}}$, $j = \boldsymbol{n}/\boldsymbol{i}$
1d or 2d to 3d:	$i =$ the smallest factor $\geq \sqrt[3]{\boldsymbol{n}}$, $j =$ the smallest factor $\geq \sqrt{\boldsymbol{n}/\boldsymbol{i}}$, $h = \boldsymbol{n}/(\boldsymbol{i} \times \boldsymbol{j})$ For 1d prime $\boldsymbol{n}$, the 2d and 3d representation will be a long string length $\boldsymbol{n}$.
3d to 2d	The 2d represention consists of $\boldsymbol{h}$ horisontal slices (levels), $\boldsymbol{i} \times \boldsymbol{j}$, one below the other, starting from the top, i.e. a series of horisontal cross-sections through the 3d volume (see figure 32.9.1).

2d and 3d to 1d This will be represented a nornal 1d space-time patterns, were the cells are indexed $\boldsymbol{n-1}$ to 0 from left to right. The cell scale is automatically reducd to 1 pixel.



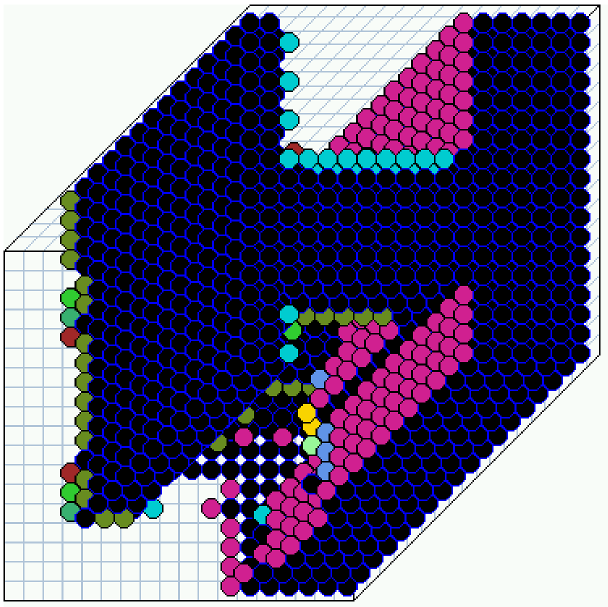


Figure 32.7: A 3d network shown in 2d by toggling 3d to 1d to 2d on-the-fly. *above*: The 3d space-pattern, 18x18x18. The projection is axonometric seen from below, as if looking up at the inside of a cage. Cells are shown colored according to neighborhood lookup by default for a clearer picture (instead of by value: 0,1). *left*: The 2d represention consists of **18** horisontal slices, one below the other, starting at the top, i.e. a series of horisontal cross-sections through the 3d volume. This is a k=*7* (nearest neighbor) 3d CA, with the totalistic code 1110100. The network was started with a random initial state, and settled to this semi-stable pattern.

32.9.2 tog 2d-2d+time

t..tog 2d-2d+time: for networks being displayed in 2d, enter **t** to toggle between normal 2d, and 2d plus a time dimention, drawn as a 3d isometric projection (see figure 4.9), best seen when cells are colored according to the neighborhood (see 32.8.3). The current status is shown in the prompt, i.e. if 2d+time is active the prompt is **t..2d+time-2d**. See figure 4.9.

32.9.3 draw 2d plane

p..2d draw plane: only if 2d+time is active (see above), enter **p** to draw a 2d translucent plane within the 2d+time projection at the current time-step to highlight the isometric geometry. This plane is also drawn whenever the seed, rule/s or wiring are changed. See figure 4.9.

32.9.4 tog 3d slice

I..tog 3d slice: If a network is being displayed 3d, enter **I** to toggle between the default display method in Unix and Linux, showing each time-step as one 3d snapshot, and showing the time-step build up of successive horizontal slices or layers starting at the top, similar to 2d+time. For DOS, this is the default and only method.

32.10 frozen/filter

"Frozen" options allow visualizing the activity/stablity of cells. "Filter" options allow progressivly filtering categories of cells, starting with the background domain, to show up interacting configurations (gliders) more clearly. Note the frozen and filter options can be used together. Filtering works for 1d, 2d and 3d networks with a homogenious rule and k (including RBN), but the method relies on data from the input-entropy histogram, so the input-entropy option must be active, the default for 1d networks. For 2d and 3d networks this option would need to be set in section 31.9.

32.10.1 nor-f1-f2-bin

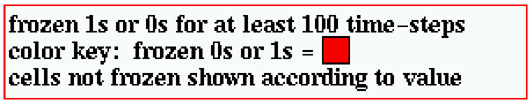
h..nor-f1-f2-bin: This on-the-fly option is a 4-way togle that allows "frozen cells", that have not changed over the last $\boldsymbol{x}$ time-steps to be highlighted (by two methods), or cells to be colored according the fraction of time they have been "on" (i.e. 1) in the same window of x time-steps, falling into preset "frequency bins", examples are shown in figure 32.10.3 (b,c,d,e).

$\boldsymbol{x}$, the "frozen generation" size, and also the number of bins and the bin boundaries can be reset in section 32.10.2. The initial defaults are, frozen generation = 20, number of bins =10, and bin boundaries equally spaced.

Enter **h** to toggle on-the-fly through the following four ways of of displaying cells, **nor-f1-f2-bin** then back to **nor**. The option in the on-the-fly window changes accordingly.

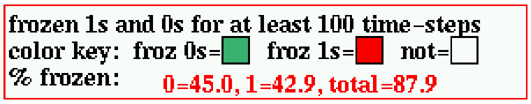
nor ... "normal" colors. If the 4-way toggy returns to **nor**, cells are shown according to the actual cell value 0 or 1, represented by a dark and a light color, even if color by neighborhood "looked up" was originaly set in section 32.8.3.

f1 ... "frozen" (method 1): frozen cells are colored red. Dynamic, unfrozen cells, are show "normal" as above. The following reminder is shown in a top right window,



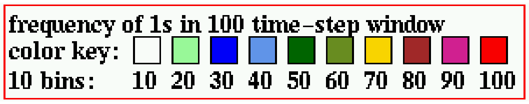
see figure 32.10.3 c.

f2 . . . "frozen" (method 2): frozen 1s are colored red, frozen 0s green. Dynamic, unfrozen, cells are colored white. The following reminder, also indicating the percentage of frozen 1s, 0s, and the total frozen, is shown in a top right window,



see figure 32.10.3 d.

bin . . . colors are assigned from a spectrum representing the fraction of time each cell have been "on" (i.e. 1) in the frozen generation time window of x time-steps, falling into a preset number of frequency "bins" and bin boundaries. The following reminder showing the bin color scheme and bin boundaries is shown in a top right window,



see figure 32.10.3 e.

The bin boundaries give the upper limit of each bin. these parameters can be reset in section 32.10.2.

32.10.2 frozen parameters

H..f-gens (now 20)/bins(10): Enter **H** to change "frozen" parameters, the default number of frozen generations, or the number of bins and the bin boundaries, relating to section 32.10.1 above. The following top right prompts are presented in turn (showing the initial defaults),

enter new 'frozen' Generation size (now 20):
reset bin boundaries, enter new total (max 10, now 10):

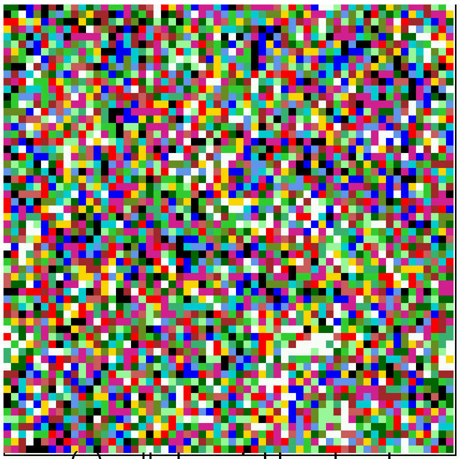
Note that the if the generation size is made less than the number bins, the number (and maximum number) of bins is reduced automaticaly, as this cannot exceed the generation size.

32.10.3 frozen generations

The frozen generation size is number of time-steps that a cell must remain unchanged to qualify as "frozen". The following top right prompt allows this to be changed,

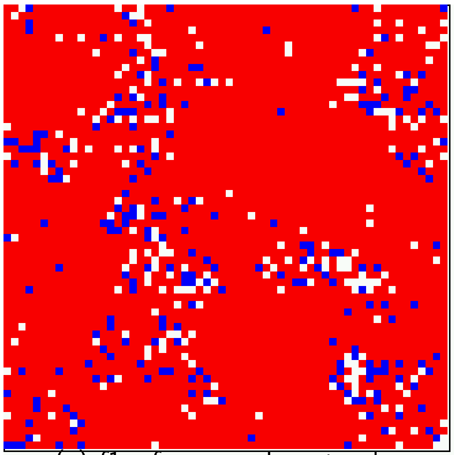
enter new 'frozen' Generation size (now 20):

Enter a new frozen generation size, which also sets the maximum number of bins. The current frozen generation size is shown in both the on-the-fly window and in this prompt. The initial default is 20 time-steps.

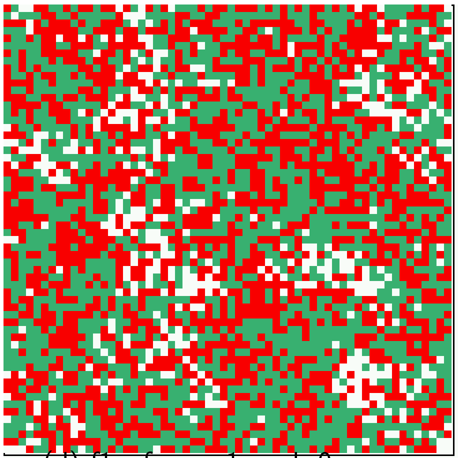


(a) cells by neighbourhood

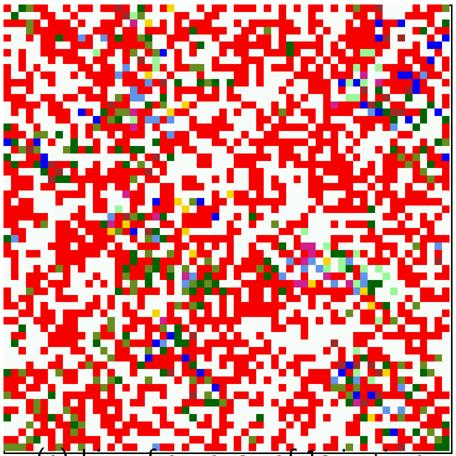
(b) cells by value



(c) f1 - frozen red, rest value



(d) f1 - frozen 1s red, 0s green



(e) bin - frequency of 1s in time window

Figure 32.8: Toggling between (a/b) cells by value and neighbourhood (see section 32.8.3), and between (b/c/d/e) cells by value and 3 different "frozen" presentations (see section 32.10.1). The network is $\mathbf{40} \times \mathbf{40}$ RBN, $\boldsymbol{k} = \mathbf{5}$ with wiring restricted to an 8 cell radius, mixed rules with canalizing inputs set to 51%. The 5 alternative presentations represent the same 2d space-time pattern.

32.10.4 frequency bins

To change the number of bins and bin boundaries, relating to section 32.10.1, the following top right prompt is presented after the frozen generation prompt in section 32.10.3,

> **...**
> **reset bin boundaries, enter new total (max 10, now 10):**

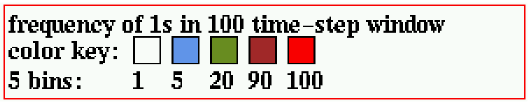
The current number of bins is shown in both the on-the-fly window and in this prompt. First enter the new bin number, which cannot be greater than the frozen generation size. Once set, a further top right prompt allows setting the bin boundaries,

> **unequal bins-u, equal-def:**

Enter **return** to set equal size bins (or as equal as possible) automatically. Otherwise enter **u** to set the bin boundaries by hand. The bin boundaries give the upper limit of each bin. The following top right prompts are presented in sequence,

> **enter bin boundary 1 (def 2):**
> **enter bin boundary 2 (def 4):**
> etc...

When cell colors are assigned according to "frequency bins" (section 32.10.1), a reminder of the current parameters is shown in a top right window. In the example below, the "frozen generation" size was set to 100, the number of bins to 5, and the bin boundaries to 1, 5, 20 and 100.



The bin boundaries must not exeed the current number of frozen generations, otherwise the following top right warning message is presented,

> **invalid bin boundaries, cont-ret:**

In this case enter **return** to reset the bin boundaries. The initial defaults are, frozen generation size=20, number of bins=10, and bin boundaries equally spaced.

32.10.5 filter/undo/all

f/F/a..filter/undo/all: Enter **f** to progressively filter space-time, **F** to unfilter in reverse order, and **a** to immediately restore the space-time pattern. Filtering works by suppressing the printing of cells that updated with reference to the current most frequent unsuppressed neighborhood. This suppresses the background domain to show up gliders more clearly (see figures 4.1.6, 32.9 - 32.11).

A dot is shown alongside the look-up frequency histogram (see section 31.9) indicating which neighbourhoods are currently suppressed (see figure 32.10. As well as progressively filtering/unfiltering, particular neighbouhoods can be filtered in isolation in section 32.14.6. On-the-fly filtering works for any single rule network, and for 2d and 3d provided that the input-entropy was selected in section 31.9.2.

For most glider rules, only a few neighbourhoods need to be suppressed to filter domains. Rules with very complicated domains, such as the $\boldsymbol{k}$=3 rules 54 and 110, must first be transformed to equivalent rules with greater $\boldsymbol{k}$ ($\boldsymbol{k}$=5 in this case) for successful filtering, which requires suppressing a number of the $\boldsymbol{k}$=5 neighbourhoods (see figures 32.11), 32.9, 32.10).

Filtering can also reveal discontinuities within or between chaotic domains (see figure 32.12.

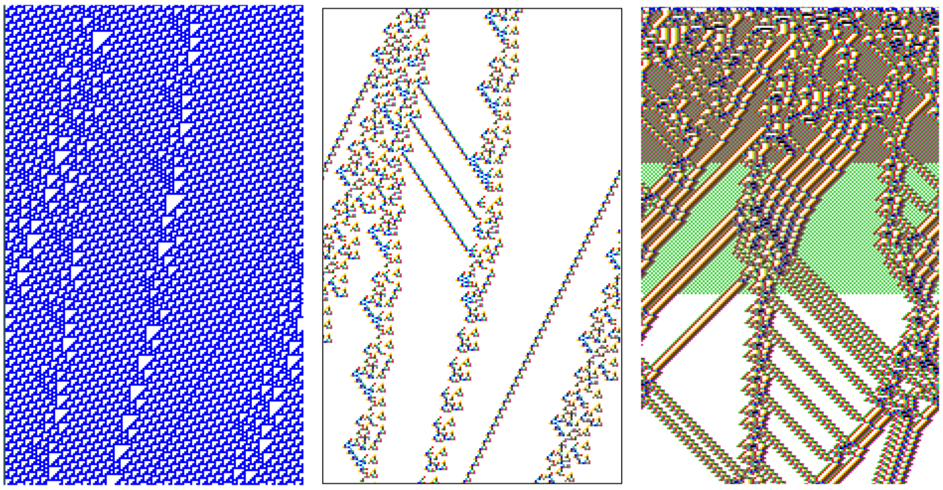


Figure 32.9: Examples of filtering space-time patterns to show up gliders more clearly. *left*: space-time patterns of the $\boldsymbol{k}$=3 rule 110, transformed to the equivalent $\boldsymbol{k}$=5 rule 3cfc3cfc, $\boldsymbol{n}$=150 (cells by value). *centre*: the same space-time pattern filtered (cells by neighbourhood lookup). *right*: Space-time patterns of the $\boldsymbol{k}$=5 rule 360a96f9. Cells are shown by neighbourhood lookup, and are progressively filtered in 2 stages. About 200 time-steps are shown in each case.

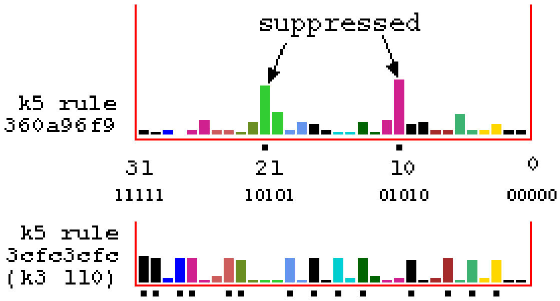


Figure 32.10: The look-up frequency histograms relating to figure 32.9. (*above*) $\boldsymbol{k}$=5 rule 360a96f9, (*below*) $\boldsymbol{k}$=3 rule 110 transformed to $\boldsymbol{k}$=5 rule 3cfc3cfc. Suppressed neighborhoods are indicated with a dot. Note that the look-up frequency histogram is presented on its side in DDLab.

32.11 analysis

These on-the-fly options relate to various methods of real-time analysis of space-time patterns.

32.11.1 tog entropy-density

s..tog entropy-density: enter **s** to toggle between showing the input-entropy and pattern density as desribed in section 31.9 and figure 31.5. The current status is given by the prompt order, i.e.

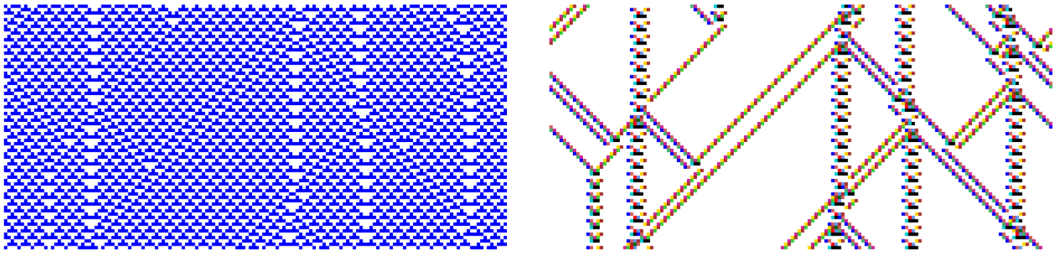


Figure 32.11: Space-time patterns from the same initial state showing interacting gliders, which are embedded in a complicated background. ***Left***: cells by value. ***Right*** cells by neighbourhood lookup, with the background filtered. The k=3 rule 54 was transformed to its equivalent k=5 rule (hex) 0f3c0f3c, n=150.

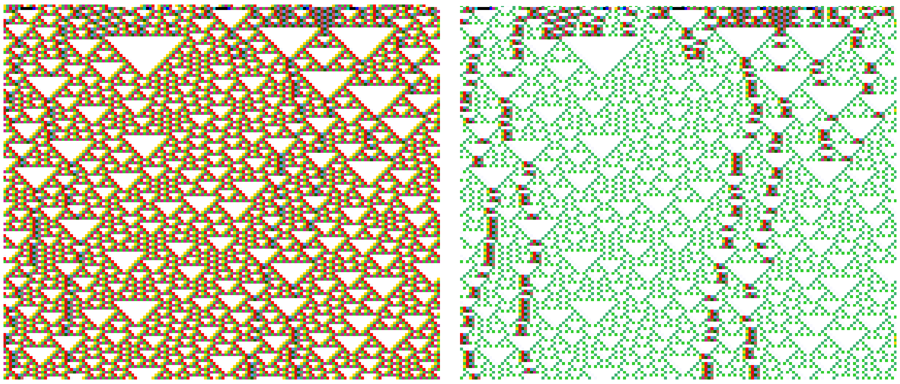


Figure 32.12: Unfiltered and partly filtered space-time patterns of k=3 rule 18 (transformed to k=5 rule 030c030c). n=150, about 130 time-steps from the same random initial state, showing discontinuities within the chaotic domain.

s..tog density-entropy indicates that **density** is current. These measures relate to a moving window of time-steps (default 10) that can be reset in section 32.11.5. For 2d and 3d networks the input-entropy and pattern density must first be set in section 31.5.

32.11.2 tog ent-fuz-both

j..tog ent-fuz-both: if input-entropy is active (see 32.11.1), **j** is a 3-way toggle to change the display between the following,

ent ... showing just the input-entropy graph.

fuz ... showing the lookup frequency of each neighborhood as a set of superimposed graphs. The colors of the graphs are the sane as the colors of the bars in the input frequency histogram described in section 31.9.2.

both ... showing both of the above simultaneously.

The current status is shown in the prompt order, i.e. if **fuz** is active the prompt is **fuz-both-ent**, if **both** is active the prompt is **both-ent-fuz**. Figure 32.11.2 gives an example.

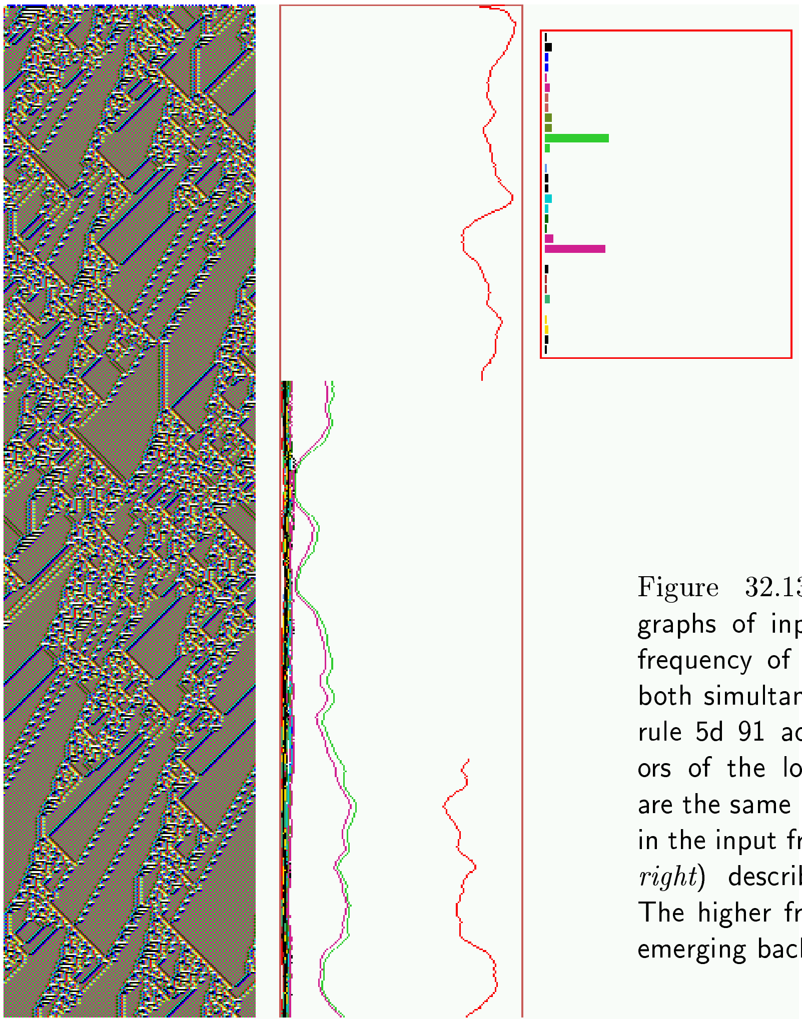


Figure 32.13: Toggling between graphs of input-entropy, the lookup frequency of each neigborhood, and both simultaneously, for the k=5 CA rule 5d 91 ac 17, n=150. The colors of the lookup frequency graphs are the same as the colors of the bars in the input frequency histogram (*top right*) described in section 31.9.2. The higher frequencies represent the emerging background domain.

32.11.3 tog hist-graph

L..tog hist-graph: if input-entropy is active (see 32.11.1), enter **L** to toggle between showing the input-frequency histogram in 2d, and expanded to 3d, with an extra time axis, shown as an axonometric projection, as in figure 31.6 (see section 31.9.2).

32.11.4 entropy/density plot

u..tog entropy/density graph: enter **u** to show/delete a density/entropy scatter plot, where the entropy (x) axis is plotted against the density (y) axis, at each time-step, in the bottom center of the screen.

Superimposed plots for a number of k=5 complex rules are shown in figure 32.14. Each rule produces a characteristic cloud of points which lie within a parabolic envelope because high entropy is most probable at medium density, low entropy at either low or high density. Each complex rule produces a plot with its own distinctive signature, with high input-entropy variance. Chaotic rules, on the other hand, give a compact cloud at high entropy (at the top of the parabola). For ordered rules the entropy rapidly falls off with very few data points because the system moves rapidly to an attractor.

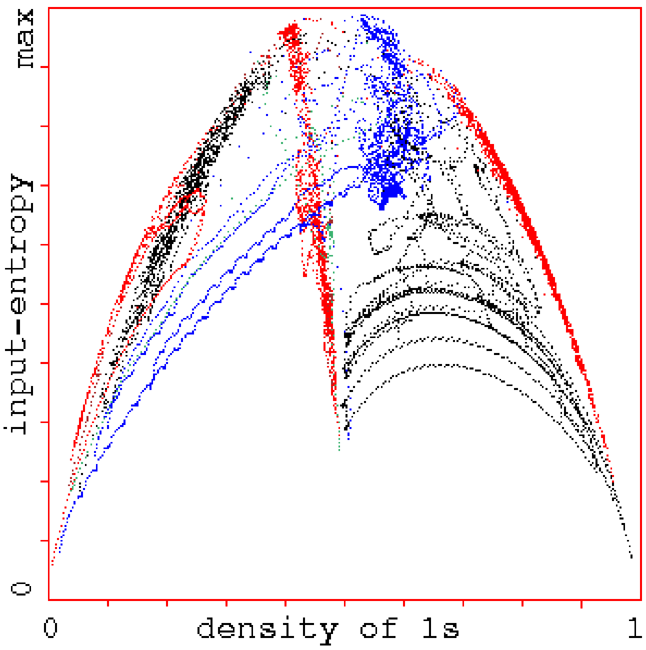


Figure 32.14: Entropy/density scatter plot[19]. Input-entropy is plotted against the density of 1s relative to a moving window of 10 time-steps. Plots for a number of k=5 complex rules (n=150) are show superimposed, each of which has its own distinctive signature, with a marked vertical extent, i.e. high input-entropy variance. About 1000 time-steps are plotted from several random initial states for each rule.

32.11.5 a-generations (analysis)

G..a-gens (now 10): Enter **G** to change the default number of analysis generations, the size of the moving window of time-steps relating to input-entropy and pattern density (options 32.11.1 - 32.11.4 above). The following top right prompt is presented,

enter new 'analysis' Generation size (now 20):

The current number of analysis generations is shown in both the on-the-fly window and in this prompt. The initial defailt is 10 time-steps.

32.11.6 return map

D..return map: enter **D** to show/delete the "return map" scatter plot in a 2d-pase plane (see also section 31.13). Each state (bitstring) $\boldsymbol{B_0, B_1, B_2, B_3 \ldots B_{n-1}}$ is converted into a decimal number 0-2 as follows,

$$\boldsymbol{B_0 + B_1/2 + B_2/4 + B_3/8 + \ldots + B_{n-1}/2^{n-1}}$$

As the network is iterated, this state value at time-step t (x-axis) is plotted against the state value at timestep t+1 (*y-axis*) in the bottom center of the screen. From a classical dynamical systems viewpoint, note the fractal structure of the resulting trajectories and attractors. An example is shown in figure 31.8.

32.11.7 state-space matrix

y..state-space matrix: enter **y** to show/delete the "state-space matrix" (see also sections 31.13 and 24.2). The state-space matrix plots each state in the space-time pattern on a 2d grid near the bottom of the screen, plotting the left half of the bit string against the right half. The y-axis represents the right $n/2$ bits. If n is odd, the extra bit is included on the left, and the grid is a flat rectangle, otherwise the grid is square. An example is shown in figure 31.7.

32.12 miscellaneous

These final miscellaneous on-the-fly options control the display of the on-the-fly key index itself, and also pausing, scrolling, stepping, and changing the speed of space-time patterns.

32.12.1 index display

X..index display: enter **X** to show/delete the on-the-fly key index itself.

32.12.2 end pause

for 1d (and 2d+time) space-time patterns only

***..tog end pause (on)**: enter * (the star key) to toggle the end pause on/off. If scrolling is not active, 1d space-time patterns (and 2d+time) are displayed in successive vertical sweeps. If the end pause on, when the itterations reach the foot of the screen further options are presented in a top right window (see section 32.14), otherwise the sweeps continue indefinitely. The current status is indicated in the prompt by **(on)** or **(off)**.

32.12.3 scrolling

for 1d pace-time patterns only

#..tog scrolling: enter # (the hash key) to toggle the scrolling on/off (see also section 31.6). 1d space-time patterns can either be displayed as successive vertical sweeps or to be scrolled upwards. The current status is indicated in the prompt by **(on)** or **(off)**. If scrolling is set, the input-entropy or pattern density plots (if set) will also be scrolled (see sections 32.11.1 and 31.9).

32.12.4 time-step pause

+..time-step pause: enter + (the plus key) to pause after each time-step (this turns scrolling off). The following top right prompt is presented,

> **time-step 6828, next-ret, reset count-r, end pause -q:** *(values shown are examples)*

Enter **return** for the next time-step, **r** to reset the time-step count to 1, or **q** to end the time-step pause.

32.12.5 slow/max speed

</>..slow/max speed: enter < (the left arrow key) to slow down the iteration speed of space-time patterns by half (eventually iteration will stop). Enter < to restore iteration back to full speed.

32.12.6 pause/no options

q/Q..pause/no-options: enter **q** to pause space-time patterns. This will also display details about the network (section 32.13.1) and present further options (section 32.14) in top right windows. To avoid showing these windows which may cover required parts of the screen, enter **Q**. Thereafter **q** will pause but no longer displays the windows, and the following botton right prompt is presented instead,

ops-Q cont-ret:

Enter **Q** to restore the original pause method, where **q** will pause space-time pattern and display further option windows, or **return** to continue space-time patterns.

32.13 Interrupting space-time patterns

If the space-time pattern itteration is interupted on-the-fly with **q** or if "end pause" is set in section 32.12.2 and the space-time pattern reaches the foot of the screen, the program will pause.

Two top right windows are displayed (if not suppressed in section 32.12.6). The upper window (single rule networks only) shows rule data described in section 32.13.1 below. The second window presents a number of further space-time pattern options, described in section 32.14.

32.13.1 Rule details for space-time patterns

for single rule networks only

On pausing the space-time pattern, a top right window shows data about the current rule similar to section 16.14. Note that showing this window when pausing can be suppressed in section 32.12.6. The window is updated as rules are transformed or mutated.

An example of the data in a rule window for a $\boldsymbol{k}$=5 rule is shown below,

■·■····■ ···■■·■■ ■■·····■ ··■··■■· **2702950694 a1 1b c1 26** *(the rule)*
k5 ld=0.406 ld-r=0.812 zl=0.617 zr=0.578 Z=0.617188
C=0/5 1d=150 time-step=1597

	key to rule details
(the rule) ...	bit pattern (for $\leq$ **9**), in decimal (for $\leq$ **5**), and in hex (for $\leq$ **9**).
k5-rule ...	the neighborhood size k.
ld, ld-r ...	$\boldsymbol{\lambda}$ parameter, $\boldsymbol{\lambda}$ ratio.
zl, zr, Z ...	$\boldsymbol{Z_{left}}$, $\boldsymbol{Z_{right}}$, the $\boldsymbol{Z}$ parameter.
C=0/5 ...	the number of canalizing inputs, in this case none out a possible 5 (k=5).
3** ...	(if applicable) shows exactly which of the k inputs are canalizing (if none this is not shown), in this example there is 1 canalizing input, at neighborhood index 3.
1d=150 ...	the network dimention and size for 1d, or the equivalent for 2d or 3d, for example, **2d=40x40** or **3d=20x20x20**.
time-step ...	the current time-step number.

32.14 Further space-time pattern options

On pausing the space-time pattern, a top right window displays a range of further context dependent options (some similar to the options in section 30.5), for amending the rule, network, seed, and some other miscellaneous pause options. Note that showing this window when pausing can be suppressed in section 32.12.6.

The following is presented in a top right window,

prtscrn with/out box-p/P savescrn-V, sample:load/keep-E/K
rule:save/load/revise/rnd/trans-s/l/v/r/t, net-n canal-C
tog graph-g, flip ends-w/b, Z:higher/lower-Z/z
state:rev-e rnd/Ham/blk/orig-R/H/k/o sng:pos/neg-5/6 save/load-S/L
filter-f skip-X pause-N step-x/+ top-T no-ops-Q back-q cont-ret:
(**Z, filter-f** *single rule networks only)*

These options are explained in sections 32.14.1 - 32.14.11 below.

32.14.1 Printing/saving the space-time pattern screen

prtscrn with/out box-p/P ... enter **p** or **P** in section 32.14 to print the screen (see section 5.5), with or without this options window and the rule window in section 32.13.1. Printing, saving, and loading the DDLab screen can also be done in the usual way (but showing the screen as is), by entering the following at any time when the prompt cursor is flashing (see section 5.4),

@ - to print
> - to save
< - to load

savescrn-V ... enter **V** in section 32.14 to save the screen image, without this options window and the rule window in section 32.13.1, as a `.nat` file, DDLab's own format (see Filing, chapter 35).

32.14.2 Load/keep the rule sample

sample:load/keep-E/K ... enter **E** in section 32.14 to load, display and probe a sample of automatically classified as rule-space (a `*.sta` file) as described in section 32.5.2. Chapter 33explains more.
Enter **K** to display and probe a sample that has already been loaded.

32.14.3 revising rule/net

The following rule and network options in section 32.14, are explained below,

> **rule:save/load/revise/rnd/trans-s/l/v/r/t, net-n canal-C**
> **tog graph, flip ends-w/b, Z:higher/lower-Z/z** (**Z:** *single rule networks only)*

option . . .	*what it means*
save/load-s/l . . .	enter **s** or **l** in section 32.14 to save or load the rule as a `.rul` file (see Filing, chapter 35) For mixed rule networks this will apply to the rule at cell index 0 only.
revise-v . . .	enter **v** in section 32.14 to revise the rule. For mixed rule networks the following prompt is presented below the top right window to select the cell index, **enter cell index, 149-0:** *(for example)* Then the rule is selected in a lower right window as described in chapter 16.
rnd-r . . .	enter **r** in section 32.14 to reset a new random rule, or all rules at random in a rule mix.
trans-t . . .	enter **t** in section 32.14 to transform the rule as described in chapter 18. For mixed rule networks this will apply to the rule at cell index 0 only (section 18.7), but canalizing inputs can be set for the whole network (see chapter 15).
net-n . . .	enter **n** in section 32.14 for the many network architecture options described in chapter 17, including viewing, revising and filing, the Derrida plot (chapter 22, and learning (chapter 34).
canal-C . . .	enter **C** in section 32.14 to change the canalizing inputs for the rule in a single rule network (section 18.7) (except the options are presented in a lower right window), or for the whole network in a mixed rule network (chapter 15).
top graph-g . . .	enter **g** in section 32.14 to show the network graph which can be rearanged in the many ways described in chapter 20, after which an additional version of the space-time graphics will be run according to the graph layout and node sizes, with corresponding presentation flexibility. Section 32.15 gives further details. Enter **g** to deactivate graph space-time patterns if active.
flip ends-w/b . . .	enter **w** or **b** in section 32.14 to flip the output of the all 1s or 0s neighborhood in a single rule, or for all rules in a rule mix.
Z:higher/lower-Z/z . . .	for single rule networks only, enter **Z** in section 32.14 to progressively force the Z-parameter higher, or enter **z** to force it lower, as in sectiona 32.4.3 and 16.9.3.

32.14.4 revising the seed

The following network state (seed) options in section 32.14, are explained below,

> **state: rev-e rnd/Ham/blk/orig-R/H/k/o sng:pos/neg-5/6 save/load-S/L**

option ...	*what it means*
rev-e ...	enter **e** in section 32.14 to revise the seed. The seed options, described in chapter are 21, are presented in a lower center window.
rand-R ...	enter **R** in section 32.14 for a new random seed.
Ham-H ...	enter **H** in section 32.14 for a new random seed with a set Hamming distance (the number of bits that differ) from the original seed. The following top right prompt is presented, **enter % ramdom Hamminig change to current state:**
blk-k ...	enter **k** in section 32.14 for a random central block of cells, the remainder set to 0. The block size was specified in section section 21.4.
orig-o ...	enter **o** in section 32.14 to restore the original seed.
sng:pos/neg-5/6 ...	enter **5** in section 32.14 for a positive singleton seed, a central cell set to 1, the remainder set to 0. Enter **6** for a negative singleton seed.
save-S ...	enter **S** in section 32.14 to save the seed or the current state. The top right following prompt is presented, **save: original seed-s, current state-c:** This will be saved as a `.eed` file, (see Filing, chapter 35).
load-L...	in section 32.14 enter **L** to load a new seed, as a `.eed` file, (see Filing, chapter 35).

32.14.5 miscellaneous pause options

The following miscellaneous pause options in section 32.14, are explained below,

filter-f skip-X pause-N step-x/+ top-T no-ops-Q back-q cont-ret:
(**filter-f** *single rule networks only)*

32.14.6 finer control of filtering

filter-f: for a single rule network only, enter **f** in section 32.14 for finer control of filtering than in section 32.10.5. The following top right prompt is presented (for example, for the k=5 CA 36 0a 96 f9 filtered twice in section 32.10.5),

filtered: 21 10 (*if unfiltered* **none** *shown here*)
filter/undo enter +/- k-index (31-0), max-m reset-r:

The top numbers (21 10) indicate the neighbourhood indices that are currently filtered, otherwise **none** is indicated. Specific neighbourhoods can be filtered and unfiltered. For example, to filter neighbourhood index 26 enter **+26**, to unfilter neighbourhood index 21 enter **-21**. Enter **m** to filter the current most frequent unsuppressed neighborhood. Enter **r** to reset, i.e. unfilter all neighborhoods. Any change is immediately indicated in the top line of the prompt. Enter **return** to accept. On resuming space-time patterns the changes will be applied.

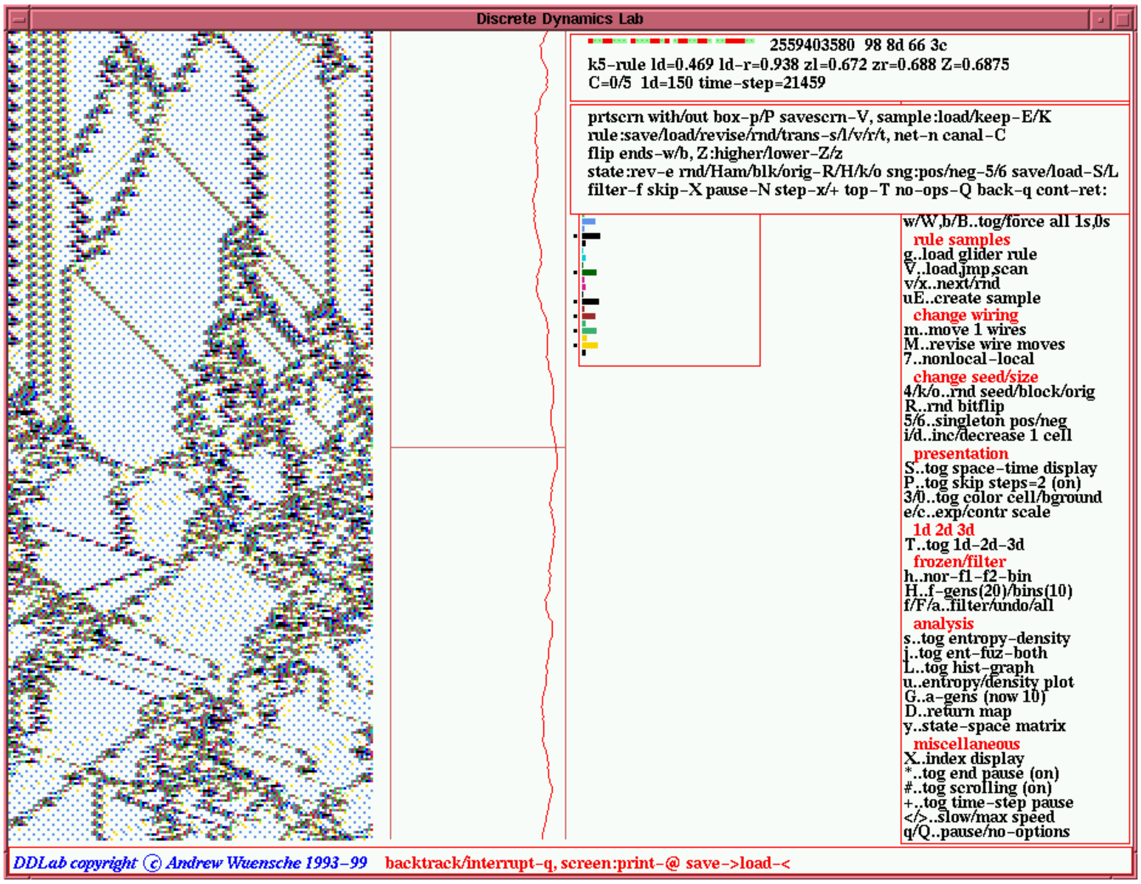


Figure 32.15: Skipping time-steps in a 1d CA, n=150, k=5 rule 98 8d 66 3c. The time-step skip was set to 2, and activated at about the mid point of the space-time pattern. The space-time pattern was also filtered and expanded to 2 pixels.

32.14.7 Skipping time-steps

skip-X: enter **X** in section 32.14 to change the number of time-steps to be skipped with on-the-fly option

> **P..tog skip steps=1 (off)** *(for example)*

in section 32.8.2. The default is **steps=1**, i.e. showing every second time-step. If this is changed to **steps=2**, every third time-step will be shown, and so forth. Figure 32.15 gives an example.

32.14.8 Set time-step pause

pause-N: enter **N** in section 32.14 to set a pause in the space-time pattern after a preset number time-steps. The following top right prompt is presented,

> **pause: screen full-p, after f-gens(20)-f**
> **or enter time-step (no pause-def):**

Enter **p** to set the end pause *on*, so a pause allways occurs when the screen is full, as in section 32.12.2 (only for 1d space-time patterns if not scrolling, and 2d+time), or **return** set the end pause *off*.

Enter **f** for a one time pause after the frozen generation number of time-steps set in section 32.10.3 (default 20), or enter a number for a one time pause after that number of time-steps.

32.14.9 Step through blocks of time-steps

step-x ... enter **x** (repeatedly) in section 32.14 to step through space-time patterns in blocks of time-steps separated by a pause. The block size is the number of time-steps that were set in section 32.14.8 above. (default 20).

step-+ ... enter + (the plus key) in section 32.14 to pause after each time-step as in section 32.12.4 (this turns scrolling off). The following top right prompt is presented,

time-step 56, next-ret, reset count-r, end pause -q:
(values shown are examples)

Enter **return** for the next time-step, **r** to reset the time-step count to 1, or **q** to end the time-step pause.

32.14.10 Start time-steps at top

top-T: enter **T** in section 32.14 to continue 1d (and 2d+time) space-time patterns starting at the top of the screen. This also resets the time-step count.

32.14.11 Quit and further options

back-q: enter **q** in section 32.14 to quit space-time patterns and backtrack. The following top left prompt is first presented,

graphics-g image:prt/save/load-p/s/l ops-o
seed-e net-n back-q cont-ret:

Enter **q** to backtrack to previous options, **return** to continue the space-time pattern, or select an option below,

option ... *what it means*

graphics-g ... enter **g** to alter the graphics setup described in section 6.3.

image:prt/save/load -p/s/l ... enter **p**, **s** or **l** to print, save, or load the DDLab screen image as described in section 5.5, but without the top left prompt window in this section (see also the **prtscrn with/out box-p/P** in section 32.14.1.
Printing, saving, and loading the DDLab screen can also be done in the usual way (but showing the screen as is), by entering the following at any time when the prompt cursor is flashing (see section 5.4),

@ - to print
> - to save
< - to load

ops-o . . . enter **o** to revert to the rule and further space-time pattern options in section 32.13.

seed-e . . . enter **e** to revise the seed. The seed options described in chapter are 21 presented in a lower central window, a repeat of **seed:rev-e** in section 32.14.4.

net-n . . . enter **n** for the many network architecture options described in chapter 17, including viewing, revising and filing, the Derrida plot (chapter 22, and learning (chapter 34), a repeat of **net-n** in section 32.14.3).

32.15 Graph layout of space-time patterns

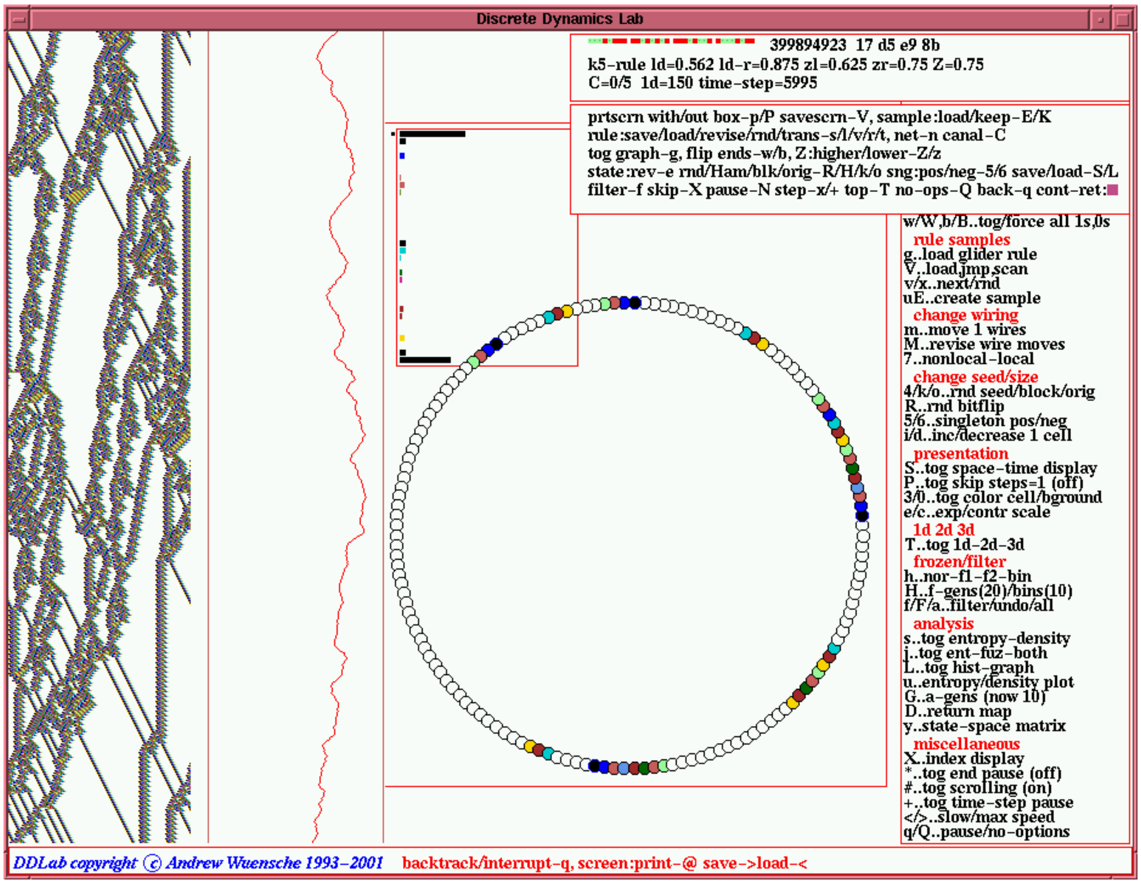


Figure 32.16: 1d CA space-time patterns shown according to the current network graph layout, in this case the default circle layout which reflects the actual geometry of a 1d CA with periodic boundary conditions. This is a complex 1d CA, n=*150*, k=5, rule 17d5e98b, with the background filtered. The normal space-time patterns on the left of the screen, and on-the-fly options, continue to function.

Space-time patterns can be shown according to the network graph layout (see chapter 20) in addition to the normal presentation described in this chapter and chapter 31. This allows enormous flexibility, as the network graph has a number of default layouts, circle, spiral, 1d, 2d, 3d, as well as allowing dragging nodes and components, and many other options for rearanging the graph.

Enter **g** in section 32.14 to start the network graph routine in a large central window. Rearange the graph and the node sizes by the methods described in chapter 20. On exiting the graph routine, space-time patters will be shown according to the final graph layout in a large central window. The normal space-time patterns, and on-the-fly options continue to work. Note that colors will correspond to the normal space-time patterns. Some examples of are shown in figure 32.16-32.17.

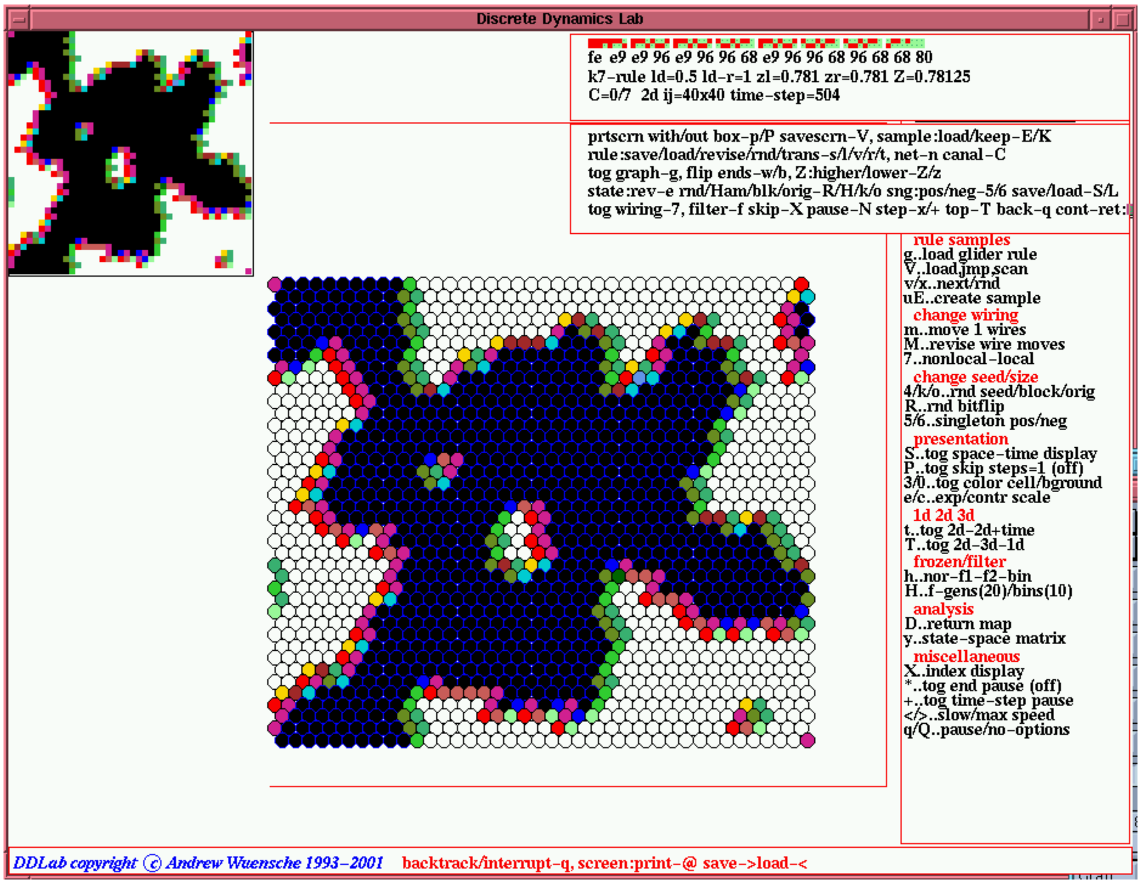


Figure 32.17: 2d CA space-time patterns shown according to the current network graph layout, in this case the default triangular grid for k=6 or 7. The network is 40x40, k=7, with a modified majority rule, totalistic code 232. The normal 2d space-time patterns, to left of the screen, and on-the-fly options, continue to function.

Chapter 33

Classifying rule space

This chapter describes methods of automatically classifying CA rule-space[19] to distinguish ordered, complex and chaotic rules, including creating, displaying, sorting, analysing and probing the sample. Unsorted and sorted 1d CA k=5 sample files, `test5.sta` and `test5ss.sta`, will be used as examples.Three larger sample files (for k=5, 6 and 7, `five5ss.sta`, `six5ss.sta` and `sev5ss.sta`) will be used to illustrate the classification. The files are included with DDLab. As well as showing the distribution of rule classes in rule-space, the method (illustrated in figure 2.4) is able to identify rules that support interacting gliders and related complex dynamics[19].

The method works by generating rules at random, then keeping track of the input-entropy for a moving window of x time-steps (default x=10, set in section 32.11.5), i.e. at each time-step the input-entropy is taken on the preceding block of x time-steps including the current time-step. To automatically classifying rule-space, the mean input-entropy is plotted against the standard deviation of the input-entropy for successive random rules, were the position of the rule on the plot indicates the rule class.

33.1 Setting parameters

To automatically create a rule sample, or run a test, first set up any CA with the desired parameters of n and k, for example a 1d CA n=150 and k=5, and run its space-time patterns (see chapter 32). For 1d the cell-scale should be to set to 1 pixel in section 7.3 and scrolling should be off in section 32.12.3. The input-entropy plot should be active, the default for 1d, but not for 2d and 3d (see section 31.9 and 32.11.1).

While the space-time patterns are iterating, select on-the-fly option **uE..create sample**, enter **u** followed by **E**. The initial **u** initiates an entropy-density plot (section 32.11.4). If this plot is current, the on-the-fly prompt is **E..create sample**, so just enter **E**. The following prompts are presented in turn in a top right window (enter **q** to backtrack or exit),

> **sample: automatic-a, test-def:**
> **specify start - end time-step of entropy record**
> **start (def 30): end (def 430): sample size for each rule (def 5):**

option ...	*what it means*
automatic-a ...	Enter **a** to create an automatic sample (section 33.3).
test-def ...	Enter **return** for a test (section 33.2), showing the measures and the entropy/density scatter plot for various rules.
start (def 30): ...	Enter the time-step when measures should start. By not including the initial time-steps, the dynamics is allowed to settle into typical behaviour before measures are recorded.
end (def 430): ...	Enter the end time-step when space-time patterns stop and measures end. The measures are recorded over *end-start+1* time-steps.

33.2 Running a test

If **return** was entered in section 33.1, a test run will be made on the current rule, for example as illustrated in figure 33.1. Space-time patterns will be run from a number of initial states (set in section 32.11.5) for the number of time-steps set in section 33.1. After each run, various measures including the varience and standard deviation of the input-entropy are shown in a top right window. At the same time the entropy/density scatter plot (section 32.11.4) is drawn in a lower center window. The plot shows distinctive signatures for complex rules, and is drawn in alternating colors for successive runs. When the runs are complete, the following top right window presents data and options (for example),

run=5/5 k5-rule=(dec)3181533384 (hex)bda258c8
mean=0.618633, average dev=0.159212, standard dev=0.182426
varience=0.042649, skewness=-0.082958, kurtosis=-0.861457
automatic-a, next rule: glider-g same-s random-def:

The top line shows the current rule in decimal (for $k \leq 5$) and in hex (for $k \leq 7$). If single cell input-entropy was set in section 31.10 the cell index will be indicated, for example **single=22**. The next 2 lines show the various measures. The last line shows prompts as follows,

option ...	*what it means*
automatic-a ...	Enter **a** to create an automatic sample (section 33.3).
next rule: glider-g ...	Enter **g** to test a random glider rule (see section 32.5.1).
same-s ...	Enter **s** to re-test the same rule.
random-def ...	Enter **return** to test a random rule.

33.3 Creating a rule sample

If **a** is entered in sections 33.1 or 33.2, a file name for the sample (`*.sta`) will first be selected in a top right window (see Filing, chapter 35), followed by a top right prompt to append the data to an existing file, or start a new file ,

my_sta.sta: append data-a, new file-def, add + to pause: *(for example)*

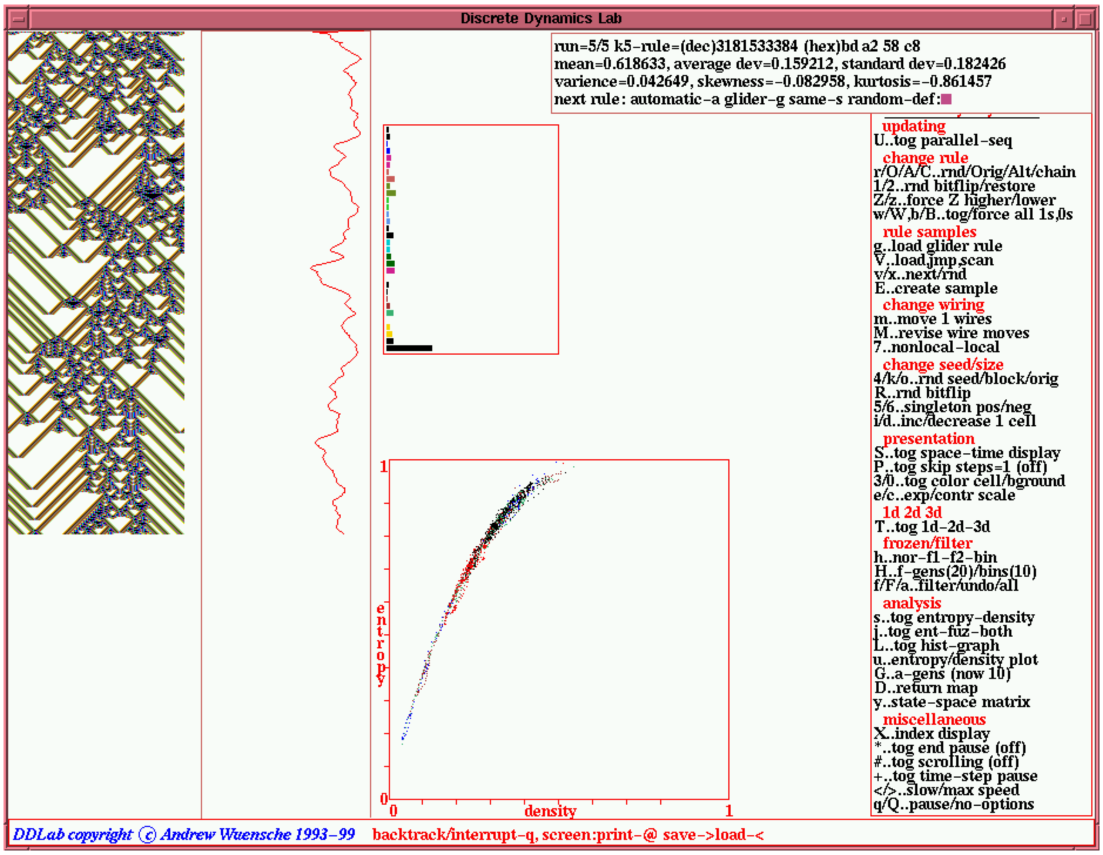


Figure 33.1: Running a test with various measures including varience and standard deviation of the input-entropy, show in a top right window. This example for a 1d CA, k=5 rule is bd a2 58 c8, n=150. At the same time the entropy/density scatter plot (section 32.11.4) is drawn in a lower center window. The plot shows distinctive signatures for complex rules, and is drawn in alternating colors for successive runs. The default parameters (start 30, end 430, size 5) were used, meaning the measures were started at time-step 30 and ended at 430, i.e. they were taken over 401 time-steps, and the number of runs from different random initial states was 5.

To pause after each new rule, enter or add +, i.e. **a+** to append and pause. The automatic sample will then start, plotting the mean entropy against the standard deviation of the entropy, one point for each new random rule in the scatter plot (see figure 33.2). After the set of runs for each rule is complete, a small red square is plotted. This is replaced by a blue dot when the next rule is complete, and so on.

For each run the same data as in section 33.2 will appear in a top right window, but including a count of the rule so far, and an option to toggle the pause, both on-the-fly and after each rule is complete, for example

rule count=1672 tog pause-p *(as in figure 33.2)*

If the pause is set, enter **return** (or **p** to toggle the pause on/off) to continue with the next rule. The sample will continue indefinitely until paused with **p**, or interrupted with **q** (enter **q**

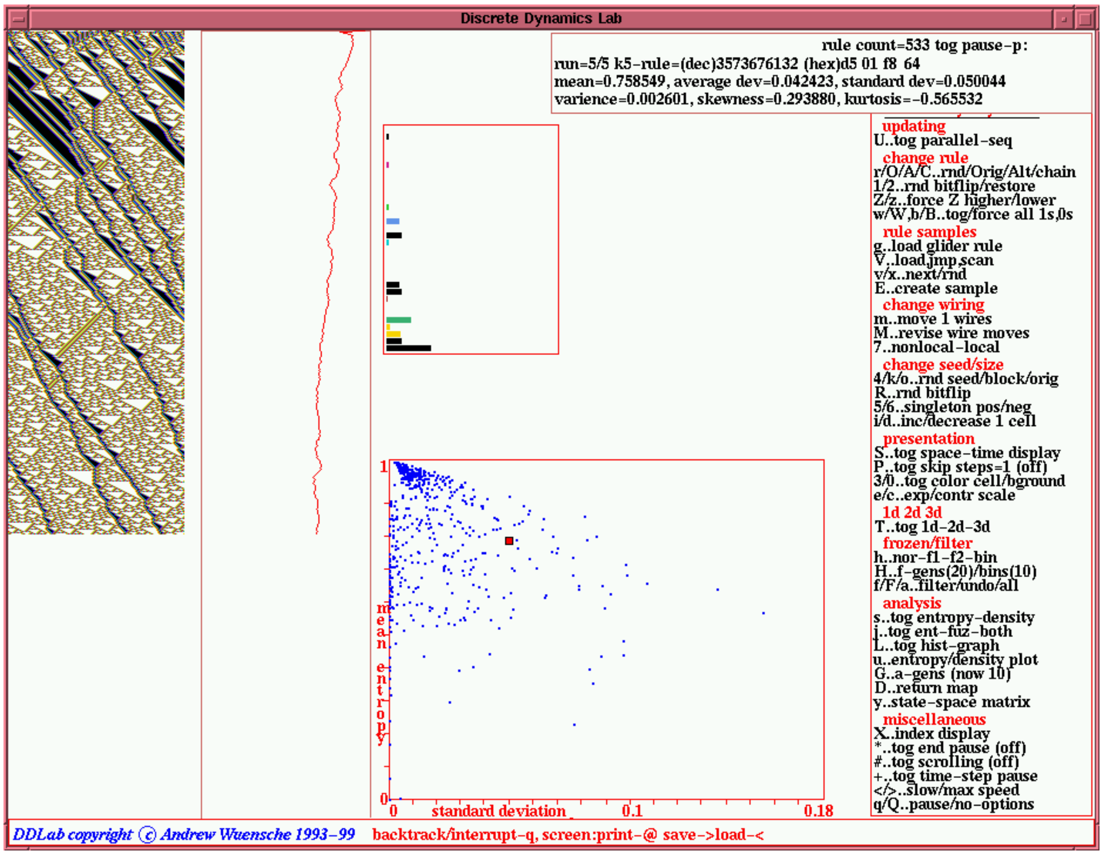


Figure 33.2: Creating an automatic sample. Various measures including varience and standard deviation of the input-entropy, and the rule count, are shown in a top right window. The sample is for a 1d CA, k=5, n=150. At the same time the mean entropy - standard deviation scatter plot is drawn in a lower center window. After the runs for each rule are complete, a small red square is plotted. This is replace by a blue dot when the next rule is complete, and so on. Note that any standard deviation above 0.18 is rounded down to 0.18

again to backtrack). If interrupted, the sample can always be continued later by appending a new sample run to an old file name.

To considerably speed up computation, turn off the space-time graphics with on-the-fly toggle **S** (section 32.8.1). Note that most on-the-fly options such as "presentation" (section 32.8) and "frozen/filter" (section 32.10) generaly work while the sample is in progress, but other on-the-fly options should be used with caution as the the consequenses are unpredictable.

33.4 Loading, sorting and displaying a sample

To load an automatic sample, enter on-the-fly option **V..load,jmp,scan**, or when interrupting space-time patterns (section 32.13) enter **E** (or **K** to "keep" the sample if previously loaded) see section 32.14.2. Once the file name is selected (see Filing, chapter 35), the following top right prompt is presented,

list: all-l pos+p, sort-s quit-q
entropy standard dev plot: Z-Z lda-L ent-def smalldots+d:

option ...	*what it means*
list: all-l ...	enter **l** to list the sample, and select a rule (sections 33.5 - 33.5.2)).
+p ...	enter **lp** to list as above, adjacent to a given mean entropy and standard deviation coordinate(section 33.5.1).
sort-s ...	enter **s** to sort the sample (below in this section).
quit-q ...	enter **q** to quit the sample and backtrack.
standard dev plot: ...	*plot (and probe) the sample in various ways, section 33.6 - - 33.6.1).*
Z-Z ...	enter **Z** to plot standard deviation against the Z parameter.
lda-L ...	enter **L** to plot standard deviation against the $\boldsymbol{\lambda}$ parameter.
ent-(def) ...	enter **return** to plot standard deviation against mean entropy.
smalldots+d ...	enter **d** (or **Zd** or **Ld**) for the above plots, but with smaller dots. For a low resolution DDLab screen (less than 640 pixels wide), the default is small dots, so the prompt reads **bigdots+d**.

Enter **l** to list the sample, or or **lp** to list just the rules at (or adjacent to) a given position on the mean entropy - standard deviation scatter plot. These option are explained in section 33.5.

Enter **s** to sort an unsorted sample. A prompt for a new file name (`*.sta`) for the sorted file will be presented (see Filing, chapter 35). The rules are sorted by decreasing standard deviation, then by decreasing mean entropy for each standard deviation. For a large sample this might take a little time.

Enter **return** or **d** to show the mean entropy - standard deviation plot (**d** shows the plot with smaller dots). Further options include probing the plot with the mouse and listing/selecting rules, (see section 33.6.1), and a 2d frequency histogram, where the vertical axis represents rule frequency. Enter **Z** or **L** for alternative plots of the Z-parameter or $\boldsymbol{\lambda}$-parameter against standard deviation. (**+d**, i.e. **Zd** or **Ld**, show the plots with smaller dots). These plots include the same further options.

33.5 Listing a sample

If **l** is entered in section 33.4 a sorted or unsorted list of the rules in the sample is presented in a top right window (see figure 33.3). For an unsorted file, the list is in the order saved.

The top line **533 k=5 rules   m-ent stan-dev** shows the number of rules in the k=5 rule sample. The rules are then listed, as many as will fit in the window, with a rule index list on the left, the rule in hex, followed by the mean entropy and standard deviation. Sets of rules separated by lines have same standard deviation. The following prompt is show at the bottom of the list,

printbox-p
more-ret jump-j quit/select-q:

option ...	*what it means*
printbox-p ...	enter **p** to print just the list window (DOS only), or the full screen in Unix and Linux (see section 5.5)

more-ret . . . if there are more rules than will fitt in the window, enter **m** to see the next batch

jump-j . . . enter **j** to junp to a new list number. The following extra prompt is presented,

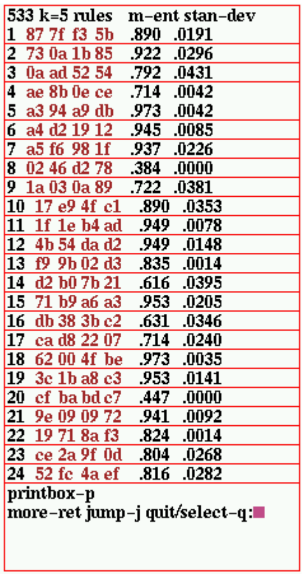
> **enter rule index 1-533:**
> **jump-j quit/select-q more-ret:**

Enter the rule index which becomes the first on the list.

quit/select-q . . . if **q** is entered, the following extra prompt is presented,

> **part list, quit-q select-s:**

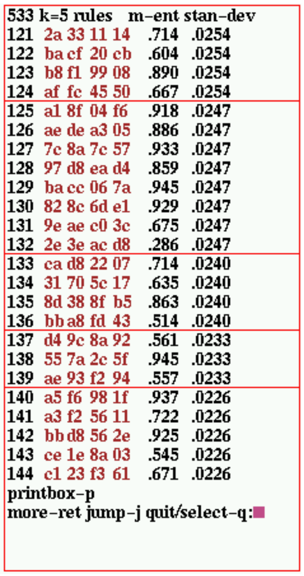
This is explained in sectio 33.5.2.



533 k=5 rules		m-ent	stan-dev
1	87 7f f3 5b	.890	.0191
2	73 0a 1b 85	.922	.0296
3	0a ad 52 54	.792	.0431
4	ae 8b 0e ce	.714	.0042
5	a3 94 a9 db	.973	.0042
6	a4 d2 19 12	.945	.0085
7	a5 f6 98 1f	.937	.0226
8	02 46 d2 78	.384	.0000
9	1a 03 0a 89	.722	.0381
10	17 e9 4f c1	.890	.0353
11	1f 1e b4 ad	.949	.0078
12	4b 54 da d2	.949	.0148
13	f9 9b 02 d3	.835	.0014
14	d2 b0 7b 21	.616	.0395
15	71 b9 a6 a3	.953	.0205
16	db 38 3b c2	.631	.0346
17	ca d8 22 07	.714	.0240
18	62 00 4f be	.973	.0035
19	3c 1b a8 c3	.953	.0141
20	cf ba bd c7	.447	.0000
21	9e 09 09 72	.941	.0092
22	19 71 8a f3	.824	.0014
23	ce 2a 9f 0d	.804	.0268
24	52 fc 4a ef	.816	.0282

printbox-p
more-ret jump-j quit/select-q:

unsorted



533 k=5 rules		m-ent	stan-dev
121	2a 33 11 14	.714	.0254
122	ba cf 20 cb	.604	.0254
123	b8 f1 99 08	.890	.0254
124	af fc 45 50	.667	.0254
125	a1 8f 04 f6	.918	.0247
126	ae de a3 05	.886	.0247
127	7c 8a 7c 57	.933	.0247
128	97 d8 ea d4	.859	.0247
129	ba cc 06 7a	.945	.0247
130	82 8c 6d e1	.929	.0247
131	9e ae c0 3c	.675	.0247
132	2e 3e ac d8	.286	.0247
133	ca d8 22 07	.714	.0240
134	31 70 5c 17	.635	.0240
135	8d 38 8f b5	.863	.0240
136	bb a8 fd 43	.514	.0240
137	d4 9c 8a 92	.561	.0233
138	55 7a 2c 5f	.945	.0233
139	ae 93 f2 94	.557	.0233
140	a5 f6 98 1f	.937	.0226
141	a3 f2 56 11	.722	.0226
142	bb d8 56 2e	.925	.0226
143	ce 1e 8a 03	.545	.0226
144	c1 23 f3 61	.671	.0226

printbox-p
more-ret jump-j quit/select-q:

sorted

Figure 33.3: Listing the rule sample in figure 33.2 with 533 k=5 rules. *left*: the unsorted sample shown in the order saved starting from rule index 1 (enter **m** to see more). *right*: the sorted sample starting rule index 121. Sorting is by decreasing standard deviation, then by decreasing mean entropy for each standard deviation. Sets of rules seperated by lines have same standard deviation.

33.5.1 Limiting the list according to plot coordinates

Entering **lp** in section 33.4, allows just the rules at (or adjacent to) a given position on the mean entropy - standard deviation scatter plot to be listed. The following top right prompts are presented in sequence,

> **mean entropy pos 0-255:** **standard dev pos 0-255:**

The y,x coordinates are plotted according to unsigned char values, so the mean entropy (0 to 1) and standard deviation (0 to 0.18) are in the range 0 to 255. Also any standard deviation above 0.18 is reset to 0.18. Enter a number (0 to 255) in each case to select a position on the plot. The same can be achieved by clicking on the plot with the left mouse buttton in section 33.6. The list will appear (similar to figure 33.3) for example,

533 k=5 pos 250x5 rad=0 *(values shown are examples)*
left mouse button to select *(applies to section 33.6)*
420 94 52 a6 ce .980 .0035 *(rules at these coordinates)*
420 2a c7 51 b6 .980 .0035

quit-q select-s rad-(max 9):

rad=0 indicates that only rules at the specific coordinate selected are shown, there may be none. **rad-(max 9)** to expand the coordinate area to include more rules, enter a radius from 1 to 9.

33.5.2 Selecting a rule from the list

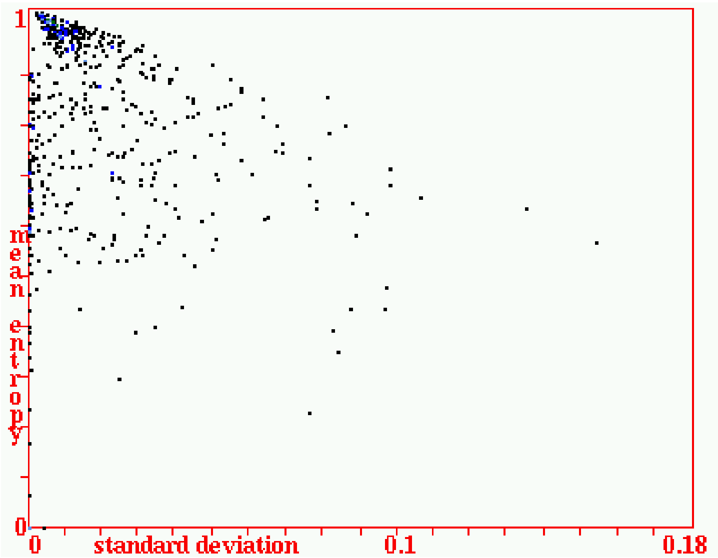
Enter **s** in section 33.5 or 33.5.1 to select a rule from the list. The following extra prompt is presented in the list window,

enter rule index 1-533: *(values shown are examples)*

Enter a rule number. The program reverts to the prompt in section 33.4. Enter **q** to quit, then return to continue with the space-time patterns of the selected rule.

Enter **q** in section 33.5 or 33.5.1 to return to the prompt in section 33.4. Note that the plots which can be selected from this prompt allow allows probing with the mouse, and also listing/selecting rules.

33.6 Plot standard deviation against mean entropy, Z or λ



from the file test5ss.sta

Figure 33.4: The mean entropy plotted against the standard deviation of the entropy. This is the test sample as in figure 33.2 with 533 1d CA k=5 rules. The sample files, unsorted (test5.sta) and sorted (test5ss.sta), are included with DDLab. A color scheme indicates frequency, clearer in figures 33.6.2 to 33.9 with a much larger samples. To select a rule anywhere on the plot, click on a point with the left mouse button to list the rule or rules, then select the rule (see sections 33.5.1 and 33.4).

If **return** is entered in 33.4 the plot of mean entropy - standard deviation is displayed in a lower center window. If **Z** or **L** is entered, alternative plots Z-parameter or $\boldsymbol{\lambda}$-parameter against standard deviation are displayed. The following top right prompt is presented,

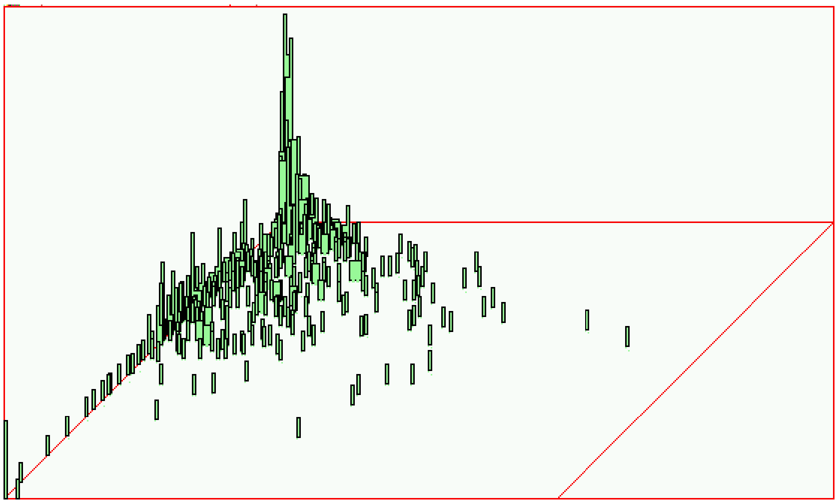
probe/select rules: point-click left mouse buttton
frequency histogram, or quit, -q

33.6.1 Probing the plot with the mouse, and selecting rules

Clicking the left mouse buttton on a point in the plot produces a top right window excatly as described in section 33.5.1, listing rules at that point, if any. The list can be expanded by setting a radius around the point to expand the coordinate area to include more rules. This is also useful to capture an isolated rule if ones aim when clicking is not so accurate. A rule can be selected from the list, and its space-time patterns run, as described in section 33.5.2.

Enter **q** (or click again) to delete the list, and click the plot again at a new position to show a new list.

33.6.2 Showing the plot as a 2d frequency histogram



from the file test5ss.sta

Figure 33.5: The 2d frequency histogram of the mean entropy plotted - standard deviation plot. The vertical axis indicates the frquency of rules falling on each quadrant of a 256x256 grid. This is the sorted test sample (test5ss.sta) as in figure 33.2 and 33.4, with 533 k=5 rules. This plot only works with a sorted sample. See figures 33.6.2 - 33.9 for 2d frequency histograms with a much larger sample.

If **q** is entered in section 33.6 the following top right prompt is presented,

533 k=5 rules: frequency histogram-f, quit-q, cont-ret:

Enter **q** or **return** to backtrack to the prompt in section 33.4. Enter **f** for the entropy - standard deviation 2d frequency histogram. The following top right prompts are presented in sequence,

freq hist, subdiv (16,32,64, etc. def=128, max-256):
save data-d, step-s, and show%-S: show freq log-l: show grid-g:

option ... *what it means*

subdiv (16,32,64, etc. def=128, max-256):
... The vertical axis of the 2d histogram indicates the frquency of rules falling on each quadrant of a grid laid over the entropy plotted - standard deviation plot. The resolution of this grid can be 16×16, 32×32, etc. up to 255×255. Enter **return** for the default 128×128, or enter **16,32,64,. . .,256** for a different resolution.

save data-d ... Enter **d** to save the height data to a an ascii file (`*.dat`), laid out in rows and columns according to the grid resolution above. For example, for a 16×16 grid, the test5ss.sta sample data (as in figure 33.2 - 33.5,is as follows,

```
125 35 2 0 0 0 0 0 0 0 0 0 0 0 0 0
22 32 19 4 1 0 0 0 0 0 0 0 0 0 0 0
20 11 8 8 2 3 0 1 0 0 0 0 0 0 0 0
17 12 7 5 5 1 1 2 0 0 0 0 0 0 0 0
18 8 8 3 2 2 3 0 1 0 0 0 0 0 0 0
21 1 10 4 2 1 2 0 1 1 0 0 0 0 0 0
24 4 2 3 2 2 1 1 1 0 0 0 1 0 0 0
10 5 9 3 3 0 0 1 0 0 0 0 0 1 0 0
4 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
3 1 1 2 0 0 0 2 1 0 0 0 0 0 0 0
2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

step-s, and show%-S ... Enter **s** or **S** to pause after each row in the 2d histogram, to see its structure in greater detail; enter **return** for the next row. If **S** is entered, the height data for successive rows is displayed in a top right window.

show freq log-l ... Enter **l** for a log scale of the vertical axis.

show grid-g ... For a grid of 64×64 or less, enter **g** to overlay the grid on the plotted - standard deviation plot.

The 2d histogram is presented in a large upper left window. Note that this plot only works correctly with a sorted sample file.

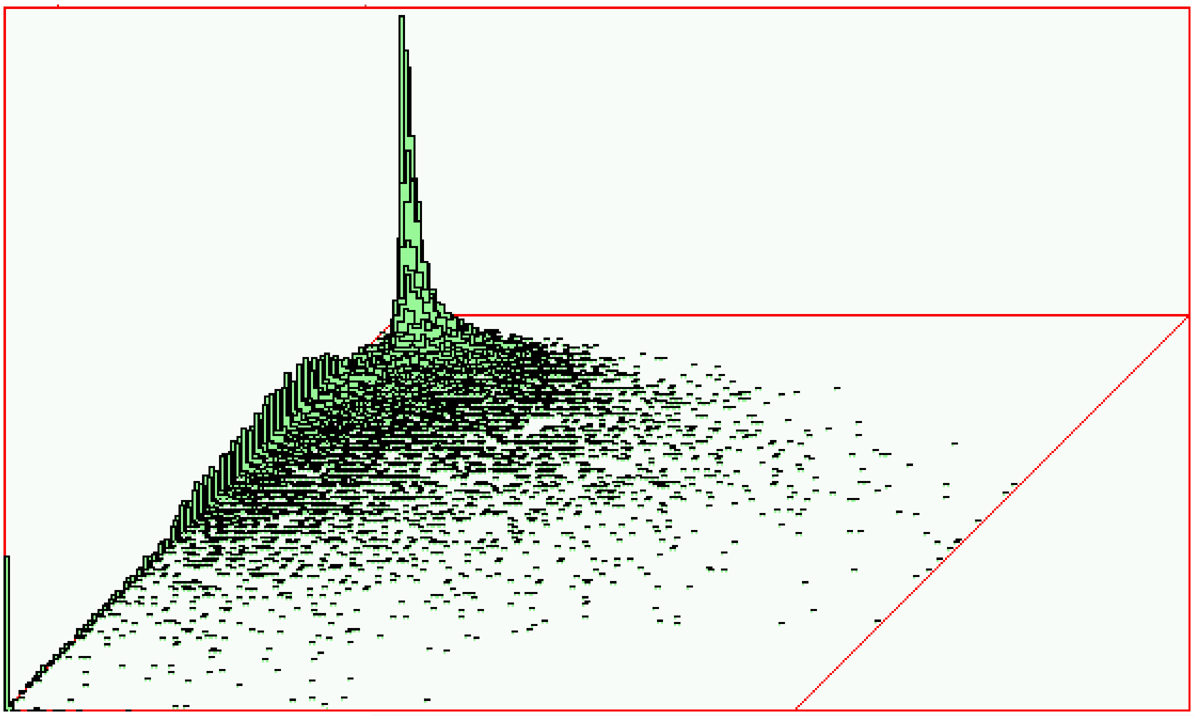
33.7 Samples files included with DDLab

As well as the 1d CA k=5 sample files, `test5.sta` and `test5ss.sta` used in the examples above, three larger sample files for k=5, 6 and 7 (`five5ss.sta`, `six5ss.sta` and `sev5ss.sta`) illustrate the classification of rule-space. The files are included with DDLab.

In these samples, for each random rule, data was gathered from 5 runs from random initial states, for 430 time-steps, discounting the first 30 to allow the system to settle, and made relative to a moving window of 5 time-steps. The three samples are shown in figure 33.6.2 - 33.9.

Looking at the k=5 2d histogram (figure 33.9), the "tower" in the upper left hand corner represents chaotic rules with low standard deviation and high mean entropy. The ridge on the left represents ordered rules with low standard deviation and a spread of lower mean entropy. Complex rules, as in figure 33.7, have higher standard deviation, and are spread out towards the right. There is a low diagonal valley between the tower and the ridge, a boundary between ordered and chaotic rules, but a gradual transition from both towards the complex rules.

Comparing the three plots for k=5,6 and 7 (figures 33.6.2 - 33.9), as k increases there is an increasing frequency of chaotic rules, a declining frequency of complex rules, and a sharp decrease in ordered rules.



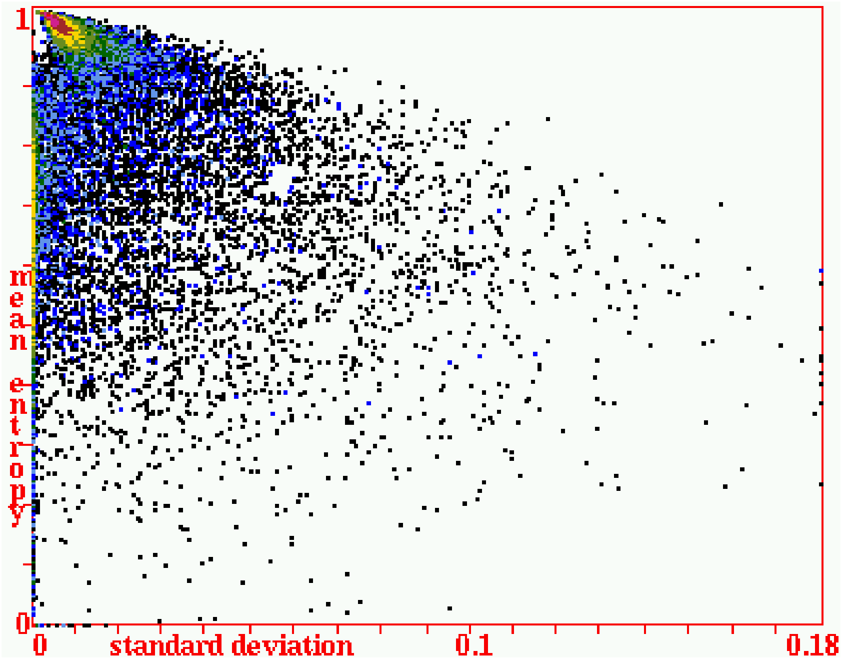


Figure 33.6: *above*: The 2d frequency histogram. The vertical axis indicates the frquency of rules falling on each quadrant of a 256x256 grid. Chaotic rules are in the "tower", ordered rules in the ridge on the left. Complex rules are spread out towards the right. *right*: The mean entropy - standard deviation plot, color coded for frequency. This is the sorted sample `five55ss.sta`, with 17680 1d CA k=5 rules.

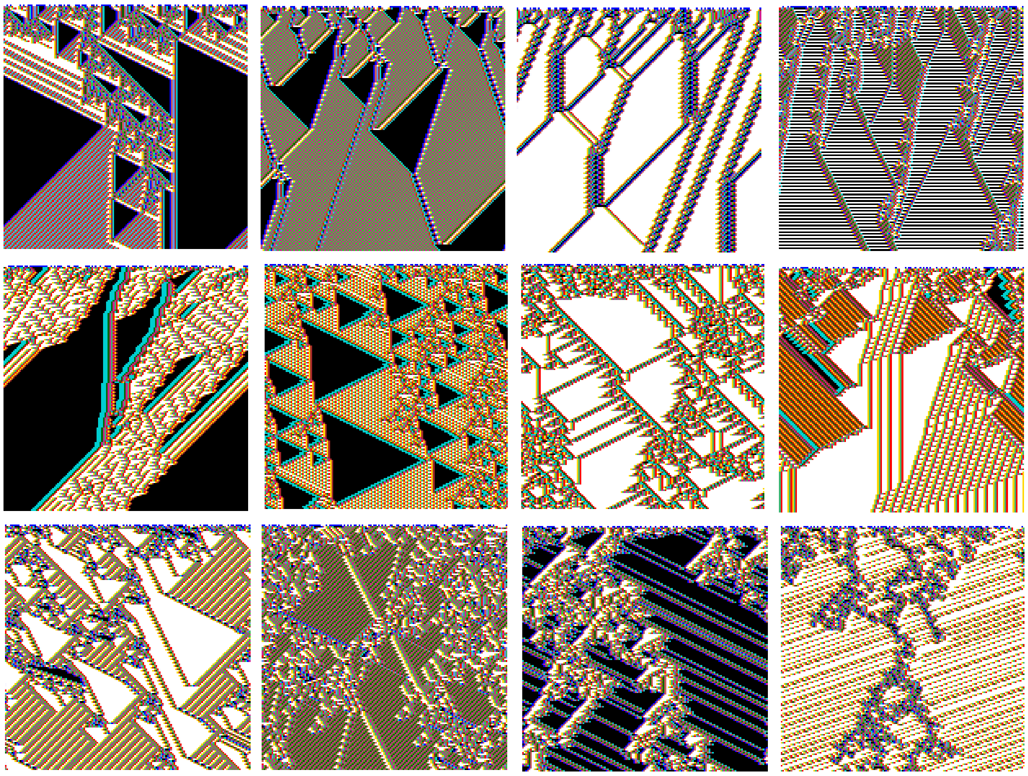


Figure 33.7: Examples of 1d CA complex space-time patterns with high standard deviation, for $\boldsymbol{k}$=5 (*top row*), $\boldsymbol{k}$=6 (*center row*) and $\boldsymbol{k}$=7 (*bottom row*). These examples, taken from the sample files `five5ss.sta`, `six5ss.sta` and `sev5ss.sta`. $\boldsymbol{n}$=150, 150 times-steps from random initial states. Cells are colored according to the neighbourhood.

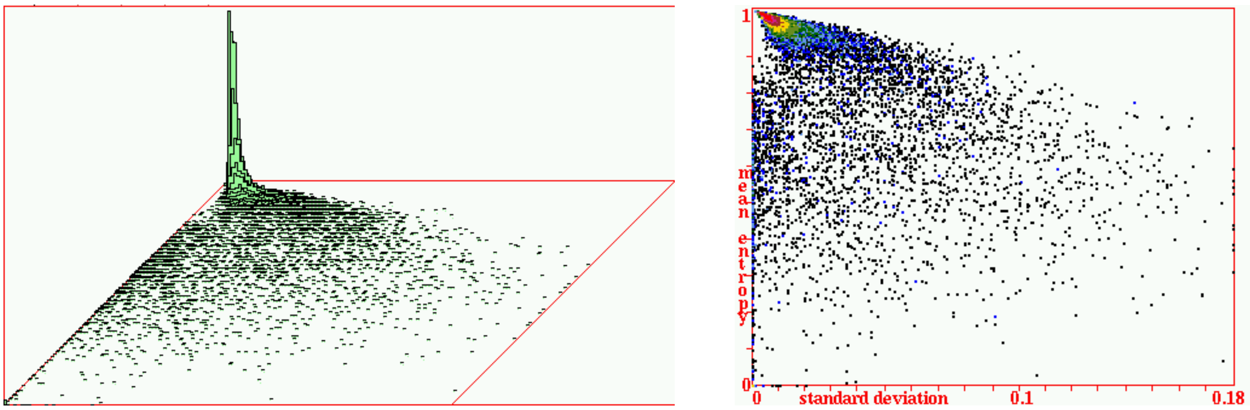
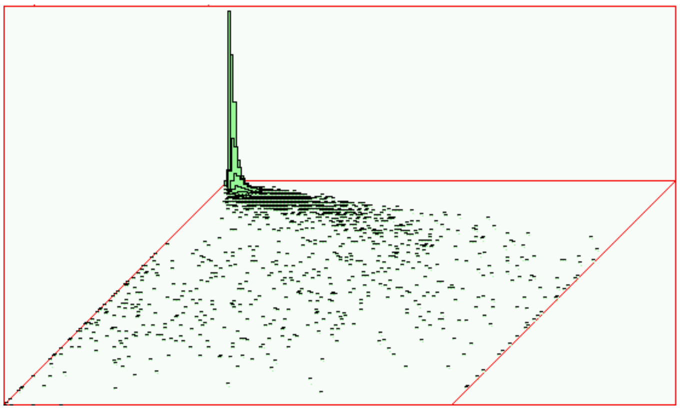


Figure 33.8: The mean entropy - standard deviation plot and histogram for the sorted sample `six55ss.sta`, with 15425 1d CA k=6 rules. Other parameters are the same as in figure 33.6.2



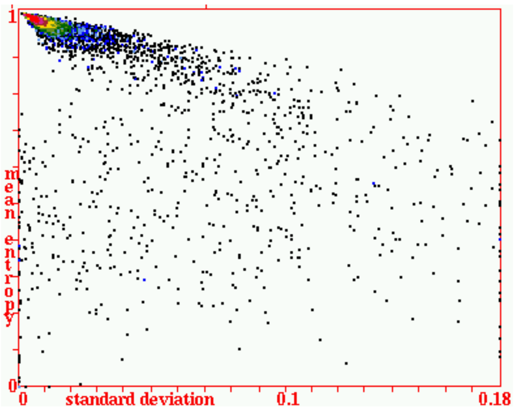


Figure 33.9: The mean entropy - standard deviation plot and histogram for the sorted sample `six55ss.sta`, with 15425 1d CA k=6 rules. Other parameters are the same as in figure 33.6.2

.

33.8 Rule sample encoding

A `.sta`) rule sample file is a binary file of an unsigned char array, recording (A) each rule, (B) its average entropy, and (C) the standard deviation of the entropy. The file records A,B,C for each rule in the sample. If the array was sorted before being saved (see section 33.4), the rules are ordered by decreasing standard deviation, then by decreasing mean entropy for each standard deviation.

The encoding of A,B,C is as follows: A (the rule-table) consists of $\mathbf{2^k}$ bytes (minimum 1 byte), B and C are both normalised to a number 0-255 and stored as one byte.

Chapter 34

Learning, forgetting, and highlighting

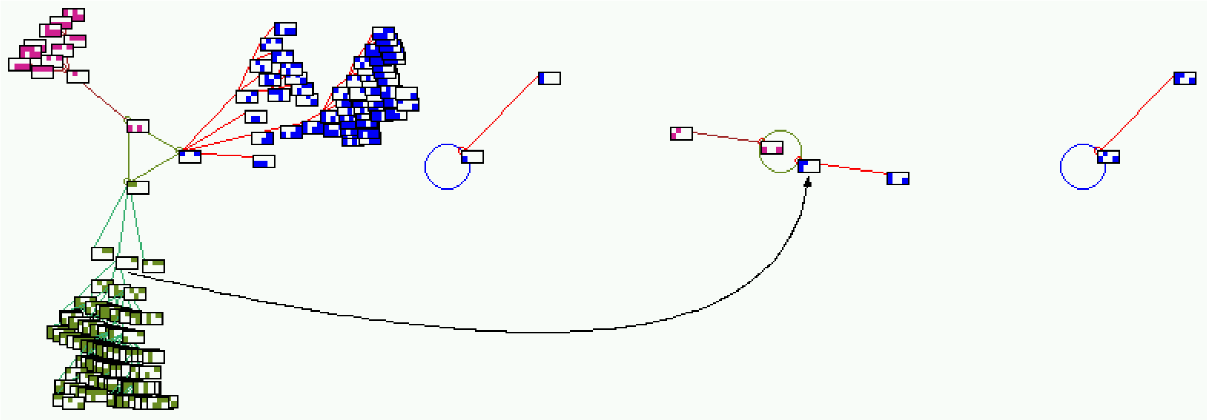
Attractors classify state-space into broad categories, the network's "content addressable" memory in the sense of Hopfield[5]. Furthermore, state-space is categorized along transients, by the root of each subtree forming a hierarchy of sub-categories. This notion of memory far from the equilibrium condition of attractors greatly extends the classical concept of memory by attractors alone[13, 15]. DDLab provides rudimentary tools for "sculpting" the basin of attraction field towards a desired category structure, by adjusting the network, flipping bits in rules or moving wires. Because any such change, especially in a small network, usually has significant side effects, the methods are not good at designing categories from scratch, i.e. solving the inverse problem or reverse engineering. A rudimentary solution to that is provided in section 18.12. However, the learning/forgetting methods might be useful for fine tuning a network which is already close to where its supposed to be.

For networks with mixed wiring and/or rules, DDLab allows one or more states to be added or deleted as pre-images of a given state. Adding or deleting pre-images is analogous to learning and forgetting. The network architecture is automatically revised to produce the required change either by moving wires or mutating rules.

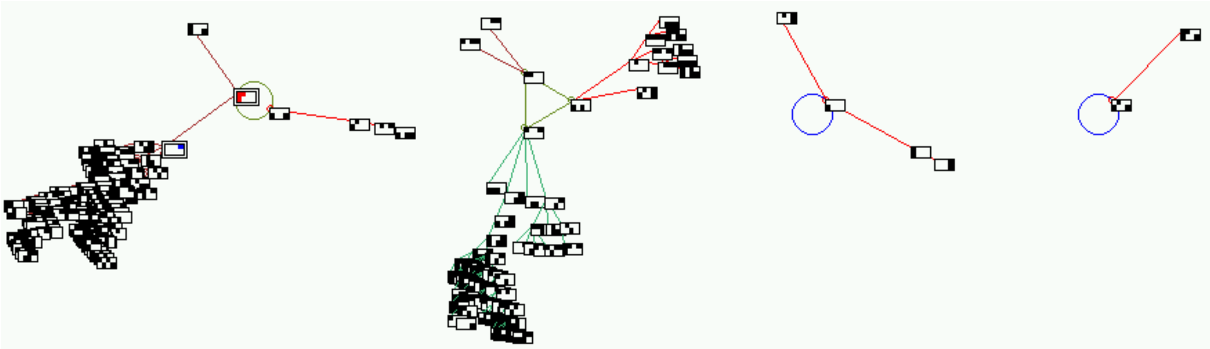
It turns out that the rule mutation algorithm is bound to learn a list of aspiring pre-images without forgetting the state's pre-existing pre-images; pre-images not on the list may also be learned. There is just one specific rule scheme mutation, which is bound to succeed. By contrast there are many alternative wire move mutations that may achieve the desired result, though success is not guaranteed, and pre-existing pre-images may be forgotten.

When applying both rule and wire learning algorithms, side effects are very almost certain to occur elsewhere in the basin of attraction, or in the basin of attraction field. The revised attractor basin can be immediately re-drawn to show the results and side affects of learning. The attractor basin can be progressively sculpted (and the network adapted) to produce a desired scheme of hierarchical categorization. In general, forgetting pre-images causes less severe side effects than learning, as minimal changes to the network architecture are required.

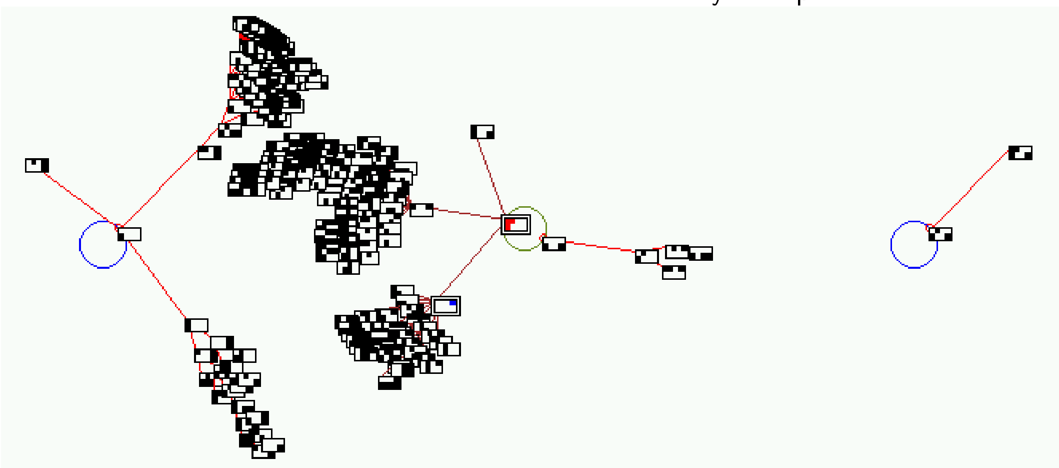
Note that learning by wire moves requires a network with non-local (random) wiring (see 11.4.1 to treat local wiring as if it where random). Mixed neighborhoods, and 2d ore 3d CA, are always treated non-local. Learning by rule mutation requires a network with mixed rules (see 12.6 to set up a rule mix where all the rules are the same).



The arrowed state has been selected as the target, and the state at the tail of the arrow the aspiring pre-image. The difference between existing image and the target is one bit (element 3), which should minimize side effects.



The result of learning by rule bitflips, and the side effects. There is just one possible solution, which must succeed. The moved state's former subtree has been mostly transplanted.



The result of learning moving wires, with greater side effects. There are multiple possible solutions, including none.

Figure 34.1: The basin of attraction field of a RBN, n=8, k=4, before and after learning. The target state and its pre-image are highlighted with a double outline. Note that the order and orientation of the basins has changed.

	3.2.1.0.	rule(hex)
7..	3 1 6 0	d2 33
6..	1 0 4 4	50 3a
5..	5 0 0 2	f6 b4
4..	1 1 2 5	7d 74
3..	5 1 7 4	b0 c5
2..	3 0 0 5	06 00
1..	4 2 3 3	9c 67
0..	3 6 5 4	12 00

the original network

	3.2.1.0.	rule(hex)
7..	3 1 6 0	d2 33
6..	1 0 4 4	50 3a
5..	5 0 0 2	f6 b4
4..	1 1 2 5	7d 74
3..	5 1 7 4	b0 c7
2..	3 0 0 5	06 00
1..	4 2 3 3	9c 67
0..	3 6 5 4	12 00

learned by rule bitflips

	3.2.1.0.	rule(hex)
7..	3 1 6 0	d2 33
6..	1 0 4 4	50 3a
5..	5 0 0 2	f6 b4
4..	1 1 2 5	7d 74
3..	5 1 7 2	b0 c5
2..	3 0 0 5	06 00
1..	4 2 3 3	9c 67
0..	3 6 5 4	12 00

learned by wire moves

Figure 34.2: The RBN networks from figure 34.1 before and after learning. *left*: the original network. *center*: the altered network, learned by rule bitflips. The rule at element 3 has changed. *center*: the altered network, learned by wire moves. One input wire to element 3 has been moved.

34.1 Highlighting states in attractor basins

The nodes corresponding to a list of states may be highlighted in attractor basins (see also section 26.2. This highlights the results of learning and forgetting, but is also useful for seeing how a particular set of states is distributed, and how the distribution changes when network parameters are altered. Highlighted nodes are shown with an outer border. The target node is shown in red and aspiring pre-images in blue. For highlighted nodes shown as bit patterns, these are the colors of 1s, otherwise 1s are shown black. Nodes other than highlighted nodes may be suppressed entirely. The highlighting feature may be used for any network including regular 1d CA.

34.2 Default attractor basin layout for learning

When using the learning algorithms for small networks, a set of possibly suitable layout parameters is available as a default. At the first "output parameter" prompt (section 24.1) select **l** to jump directly to the layout prompts. At the first layout prompt (section 25.2) select **L** to choose the default layout for learning (section 25.2.2), and set the type of node display (section 26.2), a 2d bit pattern is usually a good choice.

34.3 Selecting the wiring graphic

Whenever the network is reviewed as a 1d, circle, or 2d wiring graphic (section 17.1), learning may be implemented (option **learn-l**). Learning and highlighting is usually done while drawing basin of attraction fields (or single basins). When a field or basin is complete, the following top left prompt is presented (see section 30.4),

> **graphics-g image: prt/save/load-p/s/l, ops-o**
> **toggle: single-field-t STP-P, seed-e** *(***seed-e** *not for the basin of attraction field)*
> **net-n, back-q cont-ret:**

Enter **n** to select the top right network architecture prompts (section 17.1), which include the following options,

graphic: 1d+timesteps-1 circle-c, 2d-2: (*or* **2d+3d-3:** *for a 3d network*)

This prompt may also be reached from various other points in DDLab, described in section 17.1. When pausing space-time patterns, the prompt can be reached from the option in section 32.14.

34.4 Selecting the learn/highlight window

Enter **1**, **c** or **2** in section 34.3 above (**1** is probably the better choice) to show the wiring graphic in the lower half of the screen, and its top right prompt window (see section 17.1). Note that changes to the wiring and rules may be made directly from this prompt, which may be used in conjunction with automatic learning/forgetting.

One of the prompts reads **learn-l**. Enter **l** to open the 'learn/forget/highlight" window.

34.5 Select the target state

The initial prompts sets the "target" state, to which pre-images (i.e. direct predecessors), will be directly attached (learning), or detached (forgetting). Prompts are presented in two separate windows, one above the other. The upper window indicates that the target state is to be set,

learn/forget/highlight: set target state

The lower window presents the first of a series of prompts to select the target state,

example for a 1d network
Select target stare (1d n=14), win-w empty-e fill-f dec-d rand-r
bits1d-b bits2d-B hex-h repeat-p load-l (def-r): *(***d** *if* $n \leq 32$*)*

These prompts are the same as for selecting a seed (the initial state), described in section 21.1, but the default selection method is setting 2d bits (see sections 21.6 and 21.6.4).

34.6 Select aspiring pre-image/s

Once the target state has been selected, one or more states may be specified as its "aspiring pre-images" (i.e. direct predecessors). These states will be will be directly attached (learning), or detached (forgetting). Alternatively the states may just be just highlighted.

The set of states may be an arbitrary list, a range of decimal equivalents (if $n \leq 32$), or may be based on Hamming distance, parity, or the previous list may be repeated. The following upper prompt is presented,

pre-images: as before-b parity-p range-r hamm-h1/2 number-(def 1):
*(***range-r** *if* $n \leq 32$*)*

option ...	*what it means*
before-b ...	enter **b** to repeat the previous set of aspiring pre-images. If a list (above) was previously selected, prompts will be presented as above, but the default for each pre-image will correspond to the previous list.
parity-p ...	enter **p** to select states based on odd or even parity.
range-r ...	If $\boldsymbol{n \leq 32}$, enter **r** to specify a range of decimal equivalents.
hamm-h1/2 ...	enter **h1** or **h2** to select states based on a hamming distance of 1, or both 1 and 2, from the target state.
number-(def 1) ...	enter **return** to specify just one aspiring pre-images, or a number to specify the size of an arbitrary list of pre-images (see section 34.6.4).

Once set, the target state and list of aspiring pre-images are reviewed in a window (see #27.7).

34.6.1 Odd or even parity

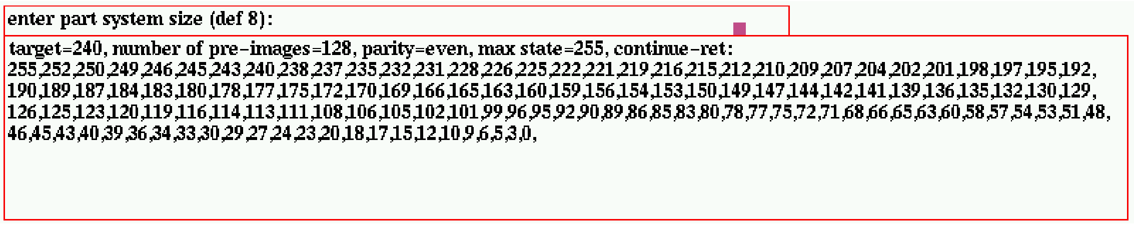


Figure 34.3: Selecting even parity in section 34.6.1, for network size, n=8. A list of the $\boldsymbol{n/2}$ decimal equivalent even parity values are displayed.

Odd and even parity signifies that the total of 1s in a bit string is an odd or even number (zero has even parity). If **p** is entered in section 34.6, the aspiring pre-images will be automatically specified as those states with the given parity relating (at most) to the first 20 cells (where the remaining cells have value 0). The following upper prompt is presented,

parity: odd-1, or even-(def):

Enter **return** for even parity, **1** odd or even parity. A further upper prompt allows the parity to relate to part only of the network, size n, where $\boldsymbol{n \leq 20}$, or part only of the first 20 network elements,

enter part system size (def 8): *(for n=8)*

If a size x, less than the system size n (or less than 20 if $\boldsymbol{n > 20}$), is entered, states with the given parity relating only to cell indices 0 to x (and where cell indices x+1 to n are equal to 0) will be selected.

34.6.2 Range of decimal equivalents

If $\boldsymbol{n \leq 32}$, enter **r** to automatically set a range of aspiring pre-images, set between two selected values (the maximum decimal equivalent that can be set is $\boldsymbol{2^{32}-1}$. The following (upper) sequence of prompts is presented to set the start and end decimal equivalents, and the "step" or increment size (default 1),

range of pre-images (def 1-8, max 255), start: end: step:
(for n=8)

34.6.3 Pre-images according to Hamming distance

Enter **h1** or **h2** in section 34.6, automatically set the aspiring pre-images based on a Hamming distance of 1, or both 1 and 2, from the target state, i.e. states that differ by just 1 bit, or by 1 bit or 2 bits, from the target state.

34.6.4 List of aspiring pre-images

Enter **return** to specify just one aspiring pre-images, or a number to specify the size of an arbitrary list of pre-images, in section 34.6. A prompt, or series of prompts to select the pre-image/s will be presented in the lower window. These prompts are the same as for selecting a seed (see section 21.1), except for the heading **Select aspiring pre-image 1**, followed by **...pre-image 2**, **...pre-image 3**, etc. for a list. The default selection method is setting 2d bits (see sections 21.6 and 21.6.4).

34.6.5 Review target state and aspiring pre-images

applies only if the maximum decimal equivalent $< \mathbf{2^{32}}$

Once the target state and list of aspiring pre-images have been set in sections 34.6 to 34.6.4 above, the decimal equivalents of the selected states, and further information, will be displayed in the lower window, as in the example for parity in figure 34.3. Examples for a range, Hamming distance, and an arbitrary list, are given below,

a range of decimal equivalents, section 34.6.4
target=240, pre-images=16, range 2-255, step=16, cont-ret:
2,18,34,50,66,82,98,114,130,146,162,178,194,210,226,242, *(for n=8)*

the set of 1-bit mutants, Hamming distance=1, section 34.6.3
target=240, pre-images=8, 1-bit mutantsrange 2-128, step=16, cont-ret:
2,18,34,50,66,82,98,114 *(for n=8)*

an arbitrary list of 5 states, section 34.6.4
target=240, pre-images=5, cont-ret:
50,167,152,88,94 *(for n=8)*

If there are more states than will fit in the window, the following prompt allows more to be displayed, or to abandon the display and continue, **more-m cont-ret:**

34.7 Learn, forget, or highlight only

Once the target state and aspiring pre-image/s have been set (sections 34.5 to 34.6.4), and reviewed (section 34.6.5), prompts are presented to learn, forget, or just highlight the aspiring pre-images (as well as the target state).

For a 1d CA (with local wiring and as single rule), or a network with 1d local wiring and mixed rules, a prompt (in the upper window) first allows the wiring to be redefined as nonlocal,

1d CA
local 1d: nonlocal wiring-w:
1d local wiring and mixed rules
local 1d, rule-mix: nonlocal wiring-w:

Enter **w** to redefine the wiring as nonlocal, allowing learning by wire moves (and removing compression in attractor basins, if set for CA, section 26.1). Otherwise the prompt continues as follow,

... highlight only-(def):

Enter **return** to highlight the selected states. The program skips directly to the prompt in section 34.11 to exit the learning routine. Highlighting is the only possibility for a network were the wiring is defined as local 1d, using the local 1d reverse algorithm (section 2.12). Note that 1d local wiring can also be redefined so that its treated as nonlocal in section 11.4.1. Highlighting on its own can be useful to show how a selected set of states is distributed in attractor basins.

For networks that do not have 1d local wiring (or if **w** was entered above) the following prompt is presented,

highlight only-h, forget-f, learn-def:

Enter **l** or **f** to learn or forget, i.e. add/remove the aspiring pre-image/s from the existing pre-image fan of the target state. When the attractor basin is redrawn according to the altered network, the target and aspiring pre-image states will be highlighted (see #27.1 and 27.**). Enter **h** to just highlight the selected states; the program skips directly to the prompt in section 34.11 to exit the learning routine.

34.7.1 Learning/forgetting by wire moves or bit-flips

If **l** or **f**, to learn or forget , was entered in section 34.7, a prompt (in the upper window) is presented to proceed by moving wires or flipping bits in rule-tables. For a single rule network, only learning/forgetting by wire moves is possible, and the prompt is as follows,

learn by wire-moves only-(def): *(or* **forget...***)*

Section 12.6 describes how to To set up a single rule network that is treated as a rule-mix.

For a mixed-rule network with local 1d wiring, only learning/forgetting by bit-flips possible, and the prompt is as follows,

learn by bit-flips only-(def): *(or* **forget...***)*

Otherwise, for networks with nonlocal wiring and a rule mix, and also networks with mixed-k which by definition has both, the prompt is as follows,

learn by wire-moves -w, bit-flips-(def): *(or* **forget...***)*

34.8 Learning/forgetting by bit-flips

The bit-flip learning algorithm[13] changes specific bits in rule-tables to make the target state the actual successor of each aspiring pre-image, instead the original successor under the original network parameters. The procedure must succeed. Pre-existing (or just learnt) pre-images can not be "forgotten", i.e. detached as pre-images of the target state, by learning more states as pre-images. However other states not on the list of aspiring pre-images may also be learned, and side effects will occur elsewhere in the attractor basin. To "forget" a pre-image just one of a set of bit-flips is required. This is chosen from the set at random. As fewer network changes are needed there will be fewer side effects.

If "**bit-flips**" was selected in section 34.7.1, the required changes to the network will be made, and the following (upper) prompt is presented,

learn/forget/highlight more-m:

Enter **m** to repeat the procedure from section 34.7, otherwise the network as it stands is accepted.

34.9 Learning/forgetting by wire-moves

The wire-move learning algorithm[13] moves a single wire at each cell position where there is a mismatch between the actual successor cell and the relevant cell in the target state. There is a choice of wires to select from the pseudo-neighborhood, and a choice of new coupling positions, some of which will correct the mismatch. The aspiring pre-image order, the pseudo-neighborhood order, and the coupling order, are all shuffled at random, and the first successful move is selected (if it exists). This may cause pre-existing (or just learnt) pre-images may be forgotten, i.e. detached as pre-images of the target state. If "just learnt" states are forgotten when learning the next pre-image, the next wire-coupling alternative in the shuffled order will be tried. If none succeed the algorithm continues with the next aspiring pre-image. On completing the learning pass, the state/s learned (and not learned) are listed by their decimal equivalents (for $\boldsymbol{n \leq 31}$ only), with the following prompt in the lower window,

failing to learning one pre-image by re-wiring
target state=73,
193,<-no
re-learn by wire-moves -w, bit-flips -b: *(or* **re-forget ...***)*
cont-ret:

tried rewiring again, successfully learning one pre-image
target state=73,
193,<-ok
success - aspiring pre-image learned *(or* **... forgotten***)*
cont-ret:

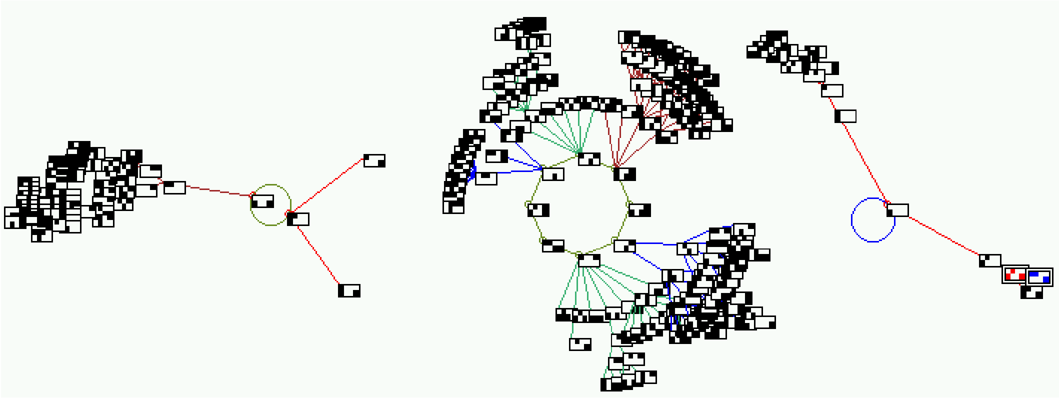


Figure 34.4: Learning one pre-image by rewiring, starting with the network in figure 34.1(*top*). The target state and its pre-image (in the last basin, on the right hand transient) are highlighted with a double outline.

learning multiple pre-images, examples for Hamming distance 1 from state 8
target state=8, 136,0<-no
24,72,12,9,10,40,<-ok re-learn by wire-moves -w, bit-flips -b: *(or* **re-forget . . .***)*

if all multiple pre-images are learned (or forgotten)
. . .
success - all aspiring pre-image learned *(or* **. . . forgotten***)*

<-ok indicates learnt states, **<-no** unlearnt states. If there are unlearnt states remaining, options are offered to try re-learn again by wire moves, or by bit-flips (which always succeeds). If **w** is selected, the last **<-no** state will be learned first, followed by the set of shuffled **<-ok** states, then the remaining shuffled **<-no** states. On completing the pass the resultant listing and prompts are repeated. Note that for a large system with a large list of aspiring pre-images, a wire-move learning pass may take an unpredictably long time.

To forget a pre-image just one appropriate wire move is required. Forgetting is generally much easier and quicker than learning. As fewer network changes are needed there will be fewer side effects. Note that in the forgetting procedure **<-ok** indicates states still attached to the target state, i.e. not forgotten, and **<-no** indicates states successfully forgotten, for example if all are **<-no** the following prompt is presented below the listed states,

. . .
success - aspiring pre-image forgotten
cont-ret:

If **return** is entered at the various prompts above, the following prompt is presented,

learn/forget/highlight more-m:

Enter **m** to repeat the procedure from section 34.7, otherwise the network as it stands is accepted.

34.10 Highlighting options

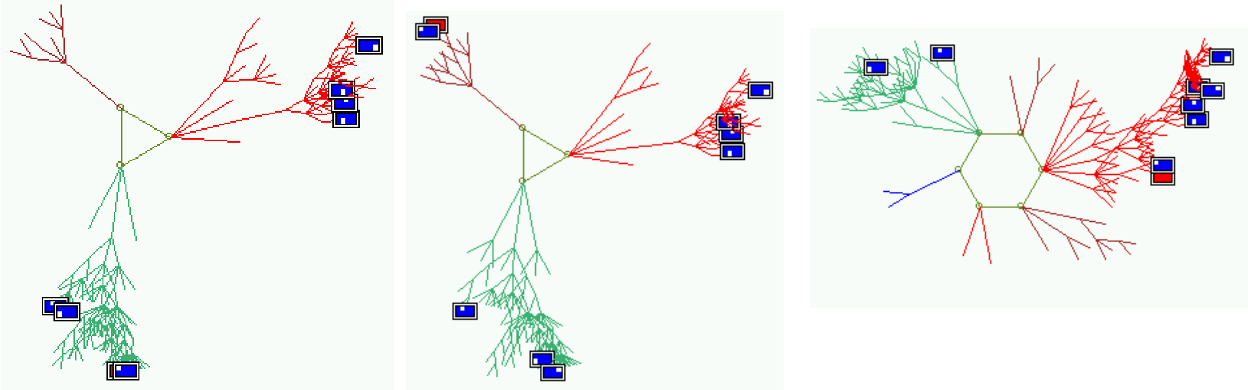


Figure 34.5: Tracking a set of states states, at Hamming distance 1 from the state all 1s, for the RBN network in figure 34.1(*top*), for a series if 1-bit rule mutations. Just one basin is show, which contains the whole set. *left*: the original network. *to the right*: a series if 1-bit rule mutations. The highlighed states have a double outline, with the "target" state shown in red, and the set of states at Hamming distance 1 in blue. All other nodes have been suppressed.

Once learning is complete, or if the "**highlighting only**" option was selected in section 34.7, the nodes representing the target state, and the set of aspiring pre-images, will be highlighted in the attractor basin. The particular form of highlighting depends on the node type selected in section 26.2,

highlight: none-n, suppess all other nodes-s:

Enter **return** to highlight selected nodes and show other nodes normally according to the node type selected in 26.2. If a 2d bit pattern was selected, the non-highlited nodes will be shown in black and white, the target state in red and white, and aspiring pre-images in blue and white, as in figure 34.4

Enter **s** to suppress the display of nodes other than highlighted nodes. Enter **n** not to highlight, but show all nodes as normal.

34.11 Learning/forgetting/highlighting complete

Following the options section 34.10, the wiring graphic and associated prompts (section 17.1) will reappear. Enter return to exit the wiring graphic, and revert to the top left "Attractor basin complete options" (section 30.4), or the pause options for space-time patterns (section 32.14).

Enter **return** to redraw the attractor basin (or restart space-time patterns) with the new (learnt) parameters. Attractor basins will be drawn with the specified highlighting. Note that after the attractor basins have been drawn, if **return** is entered again, the network will be changed by the mutations set (by default) in chapter 28. This may be what is required, for example to see how a set of highlited states is redistributed for a succession of mutated attractor basins, as in figure 34.5. However, when "sculpting" the basin of attraction field, its best to first deactivate mutation in section 28.1.

Chapter 35

Filing

DDLab has functions for saving and loading a variety of DDLab specific file types, which have default three letter extensions, for example `myrule.rul`. As DDLab was first developed as a DOS program, DOS file name conventions apply for all versions (including Unix and Linux), so a file name has up to eight characters (starting with a letter) plus a 3 character extension. This chapter lists the file types, and describes the methods for saving and loading files.

The following file types are described in more detail in other sections in the manual, including their encoding, and loading constraints due to compatibility with the current network.

Network architecture files ... (`.mix`, `.w_s`, `.r_s` and `.wrs`) Described in detail in chapter 19. Encoding: sections 19.7 and 19.7.1. Constraints that apply to loading different types of file into a given network: section 19.8.

Single rule files ... (`.rul`) Encoding: section 16.13.1. k constraints: section 16.13.2.

The network state ... (`.eed`), for the seed or initial state. Encoding: section 21.11.1. Size/dimension constraints: sections 21.10 - 21.11.

35.1 File types, default extensions and filenames

The DDLab specific file types, default extensions, and default filenames are listed below.

type	*extension*	*default file name*	
SCREEN	`.nat`	`myimage.nat`	The screen image, (section 32.14.1). Encoding: (the same for all platforms) has a homemade format too involved to explain here. A `.nat` file, saved on a given platform can be successful loaded back to the same platform, even if the screen size has been changed. If saved on one platform and loaded back to another, the results are less reliable.
K-MIX	`.mix`	`my_mix.mix`	The neighborhood-mix, (sections 8.5, 8.12.3, 19.4). Encoding: see section 19.7.1.

type	*extension*	*default file name*	
WIRING ONLY:	`.w_s`	`my_wso.w_s`	The wiring scheme (chapter 19). Encoding: sections 19.7.
RULE SCHEME ONLY	`.r_s`	`my_rso.r_s`	The rule scheme (chapter 19). Encoding: sections 19.7.
WIRING+RULE SCHEME	`.wrs`	`my_wrs.wrs`	Both wiring and wules (chapter 19). Encoding: sections 19.7.
SINGLE RULE	`.rul`	`myrule.rul`	A single rule for a given k (chapter 16, sections 30.5.1, 32.14.3). Encoding: section 16.13.1. When setting a rule in chapter 16, the last rule for a given k is automatically saved, with the filename `last[`k`].rul`, where k=neighborhood, i.e. `last5.rul`.
SEED	`.eed`	`mydeed.eed`	A state bitstring, including its size n, dimension (1d, 2d or 3d), and the i,j,h coordinates (sections 30.5.2, 32.14.4). Encoding: section 21.11.1.
DATA	`.dat`	`my_data.dat`	An ascii file for the field/basin/subtree and list of states data (sections 27.5-27.7.2, 19.5). Encoding: ascii.
PRE-HIST-DATA	`.prh`	`my_prh.prh`	Data for the pre-image histogram. (section 24.3.6). Encoding: a binary file of an unsigned long array, recording each in-degree frequency, from zero up.
HIST-DATA	`.his`	`my_his .his`	Data for various other histograms (sections 8.12.1, 17.8.13, 31.16, 31.18.3, 31.19). Encoding: a binary file of an unsigned short array, recording the height of each histogram bar, from zero up.
EXHAUSTIVE	`.exh`	`my_exhau.exh`	An exhaustive mapping, listing each state and it successor (section 29.5.1, 29.5.3, 18.1.2). Encoding: a binary file of a char array recording a list of $\mathbf{2^n}$ successor bitstrings, each in $\mathbf{2^n/8}$ bytes (minimum 1 byte), where the pre-image is the array index, from zero up.
ORDER-DATA	`.ord`	`my_order.ord`	For the sequential updating order. (section 29.7, 29.7.3). Encoding: a binary file of an unsigned short array, recording the sequential order according to the array index, from zero up.
STAT-SAMPLE	`.sta`	`mystat.sta`	The statistical sample of automatically classified rule-space, (chapter 33, sections 33.3, 33.4, 32.14.2). Encoding: section 33.8.
GRAPH LAYOUT	`.grh`	`my_grh.grh`	The layout of the network graph (section 20.2), or. the attractor metagraph (section 20.3). Encoding: a binary file of a signed short array, recording the $\boldsymbol{x, y}$ coordinates of each graph node according to the array index, from one up.

35.2 Saving and Loading

In general when saving or loading a file, the following prompt appear in a top right window. Note that changing the directory applies only to DOS,

> **SAVE** [or **LOAD**] [type] **- filename (no ext) .**[extension] **will be added**
> **change dir-d, now:** [current path] *(DOS only)*
> **list-?, quit-q, default** [filename]**:**

For example, if and loading a single rule, the prompt will appear as follows,

> **LOAD SINGLE RULE - filename (no ext) .rul will be added**
> **change dir-d, now: C:\ddlab** *(DOS only)*
> **list-?, quit-q, default myrule:**

To load (or save), enter a legal filename (without the filename extension, which is automatically added), or enter **return** to accept the default filename `myrule`. When the filename is entered, say `rule193`, the lowest line of the prompt will respond with,

> **filename=rule193.rul REVISE-q cont-ret:**

If the file is not found, the following message appears, and the program returns to the point before loading was called.

> **file error rule193.rul, cont-ret**

35.3 Changing the directory, DOS only

For DOS only, the current directory is shown in section 35.2, and can be changed by entering **d**. The following prompt appears, assuming the current directory is `c:\ddlab`,

> **current path C:\DDLAB, CHANGE drive/directory**
> **new path, (ie c:\dir\subdir):**

Enter the full new path, (i.e. c:\newdir\newsub\etc). The prompt in section 35.2 will reappear with the new path, which will stay current until changed again. If the new path has an illegal name, the following message appears,

> **cant change to** [illegal path]

35.4 List files

To list the files with the current extension enter **?** in section 35.2. A list will be displayed in a window on the right of the screen. The following prompt appears at the bottom of the list,

list again-a
more-m *(if there are too many files to fit)*
or filename
: *(the file name (without extension) can be entered here*

If there are too many files to fit, enter **m** to see more. A filename (without extension) may be selected in this window, or enter **return** and select the filename in section 35.2.

Bibliography

[NOTE] For publications by A.Wuensche refer to
`www.santafe.edu/~wuensch/publications.html`
where on line versions, abstracts, summaries and further details may be found.

---

[1] Albert,R., H. Jeong, and A-L Barabasi, (2000) “Error and attack tolerance in complex networks”, Nature, vol 406, July 2000.

[2] Conway,J.H., “What is Life?” in “Winning ways for your mathematical plays”, Berlekamp,E, J.H.Conway and R.Guy, Vol.2, chap.25, Academic Press, New York, 1982.

[3] Gutowitz,H.A., ed.,“Cellular Automata, Theory and Experiment”, MIT press, 1991.

[4] Harris,E.S., B.K.Sawhill, A.Wuensche, and S.Kauffman, “Biased Eukaryotic Gene Regulation Rules Suggest Genome Behaviour is Near Edge of Chaos”, Santa Fe Institute Working Paper 97-05-039, 1997.

[5] Hopfield,J.J. “Neural networks and physical systems with emergent collective computational abilities”, Proceedings of the National Academy of Sciences 79:2554-2558, 1982.

[6] W.Hordijk, J.P.Crutchfield, M.Mitchell, “Mechanisms of Emergent Computation in Cellular Automata”, In A.E. Eiben, Th. Back, M. Schienauer, and H-P. Schwefel (eds.), Parallel Problem Solving from Nature, Springer-Verlag, pp. 613-622, 1998

[7] Kauffman,S.A., “The Origins of Order, Self-Organization and Selection in Evolution”, Oxford University Press, 1993.

[8] Langton,C.G., (1986) “Studying Artificial Life with Cellular Automata”, Physica D, 22, 120-149.

[9] Langton,C.G., “Computation at the Edge of Chaos: Phase Transitions and Emergent Computation”, Physica D, 42, 12-37, 1990.

[10] Somogyi,R., and C.Sniegoski, “Modeling the Complexity of Genetic Networks: Understanding Multigenetic and Pleiotropic Regulation”, COMPLEXITY, Vol.1/No.6, 45-63, 1996.

[11] Wolfram,S., ed. “Theory and Application of Cellular Automata”, World Scientific, 1986.

[12] Wuensche,A., and M.J.Lesser. “The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata”, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA, 1992.

[13] Wuensche,A., "The Ghost in the Machine; Basin of Attraction Fields of Random Boolean Networks", in Artificial Life III, ed C.G.Langton, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA, 1994.

[14] Wuensche,A., "Complexity in One-D Cellular Automata; Gliders, Basins of Attraction and the Z Parameter", Santa Fe Institute Working Paper 94-04-025, 1994.

[15] Wuensche,A., "The Emergence of Memory", in "Towards a Science of Consciousness", (1996), eds. S.R.Hameroff, A.W.Kaszniak, A.C.Scott, MIT Press, 1996.

[16] Wuensche,A., "Attractor Basins of Discrete Networks; Implications on self-organisation and memory", Cognitive Science Research Paper 461, Univ. of Sussex, D.Phil thesis, 1997.

[17] Wuensche,A., "Genomic Regulation Modeled as a Network with Basins of Attraction", Proceedings of the 1998 Pacific Symposium on Biocomputing, World Scientific, Singapore, 1988.

[18] Wuensche,A., "Discrete Dynamical Networks and their Attractor Basins", Proceedings of Complex Systems '98, University of New South Wales, Sydney, Australia, 1998.

[19] Wuensche,A., "Classifying Cellular Automata Automatically; Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter", COMPLEXITY, Vol.4/no.3, 47-66, 1999.

Index