Co-evolution – with animats in pursuit-evasion

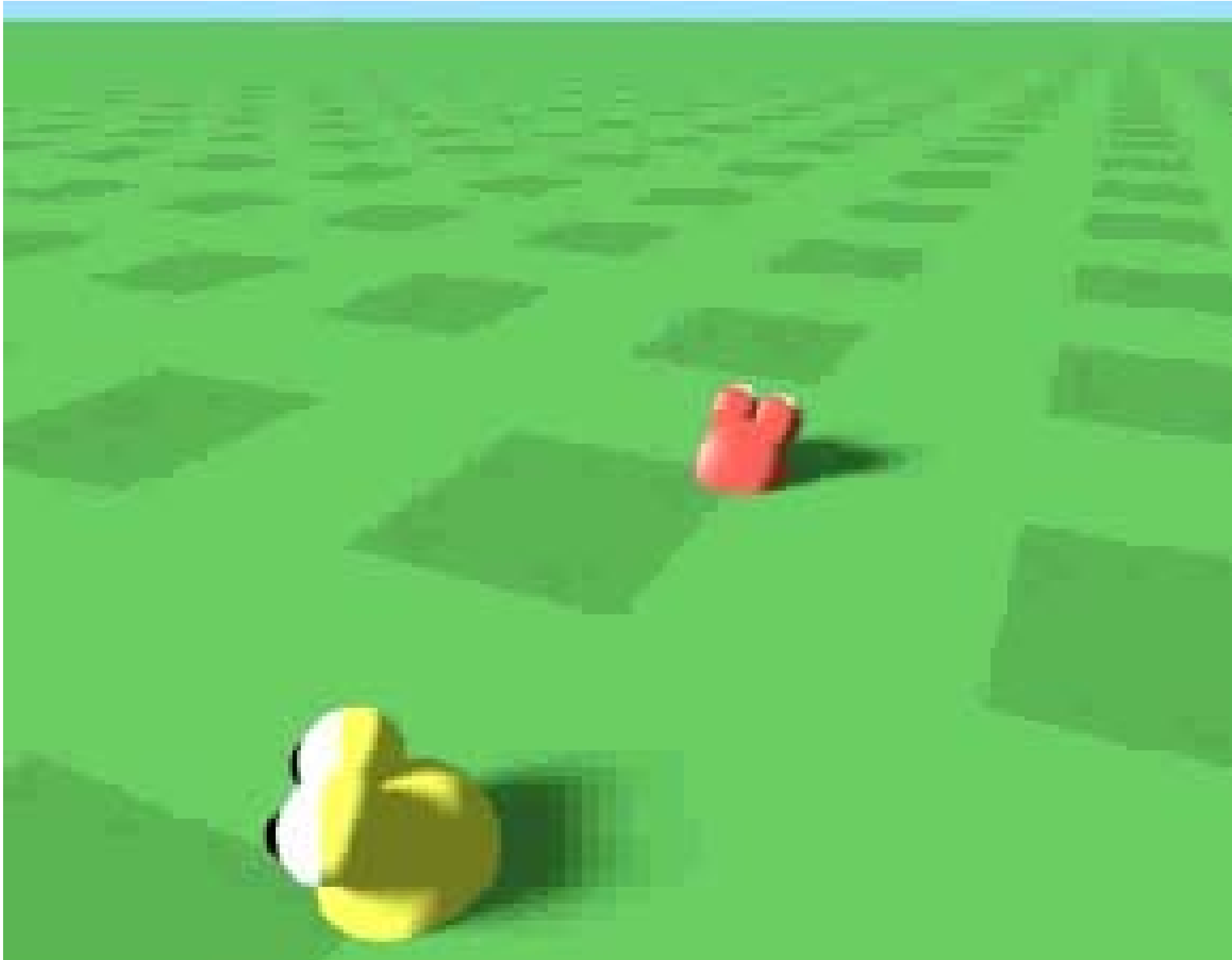
… and in an application to 'sorting networks'

Coevolution of Pursuit and Evasion

D. Cliff and G. F. Miller
``Co-Evolution of Pursuit and Evasion II: Simulation Methods and Results''. In
P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson (eds) From Animals to Animats 4
MIT Press Bradford Books, pp.506-515, 1996.

This paper, plus related ones, plus **mpegs** on
http://www.cogs.susx.ac.uk/users/davec/pe.html

Pursuit/Evasion – gen 200



Coevolution

Two (or more) species evolve in a situation where the selection pressure on one species (eg the Predators or Pursuers) depends (at least in part) on the current fitness of the other spec (Prey or Evaders) .. .. and vice versa

Arm's Race, or 'Red Queen effect'
-- you run as fast as you can yet stay in same place
(... figuratively !)

This provides very much an **implicit** fitness function rather than explicit one.

This study of coevolution

This study of coevolution

Studied in deliberately simplified environment - a 2-D infinite pla
with no walls or obstacles, just one pursuer, one evader

Animats (animal/robot) - term often used in SAB

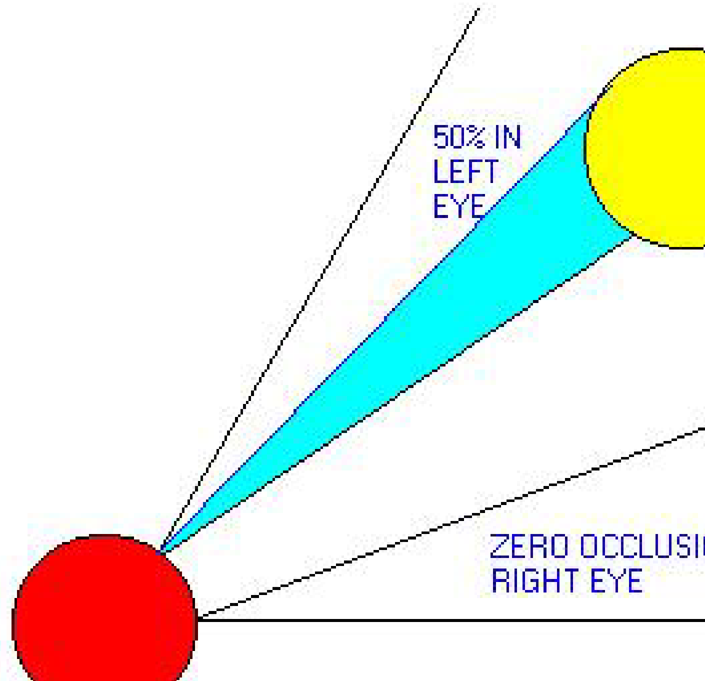
Motors:
These animats have left and right wheels. Variable forces can be
applied L and R, simple Newtonian physics

Fuel use (from limited fuel tank) proportional to square of force.
Friction acts to slow you down.

Sensors

Each animat has several (typically 2) simulated 'photoreceptors'
Position (relative to straght-ahead) and angle of acceptance (wide/narrow) is genetically specified -- and hence can co-evolve with the 'brain'

Each sensor returns proportion of of its angle-of-view which is **not** obscured by any object on horizon



Hence simulation is a very simplified version of real physics, but still has some significant element of physical plausibility

Neural Network Control System

The control system is a CTRNN
(continuous-time recurrent neural network model),
of precisely the Beer type (see previous lecture).

Fully connected ANN, with (fixed) weights and biases that are genetically specified -- ie evolved.

2 neurons connected to 'eyes', 2 to motors.

Evolution

A Genotype for any one Animat specifies -

1) the sensory morphology
2) the architecture (weights etc) of the ANN

The Genetic Algorithm evolves 2 completely distinct populations ('species')

Spatially distributed GA -- individuals in the population are spr out over a 'mating' grid, and will only mate, and replace, close neighbours on this grid.

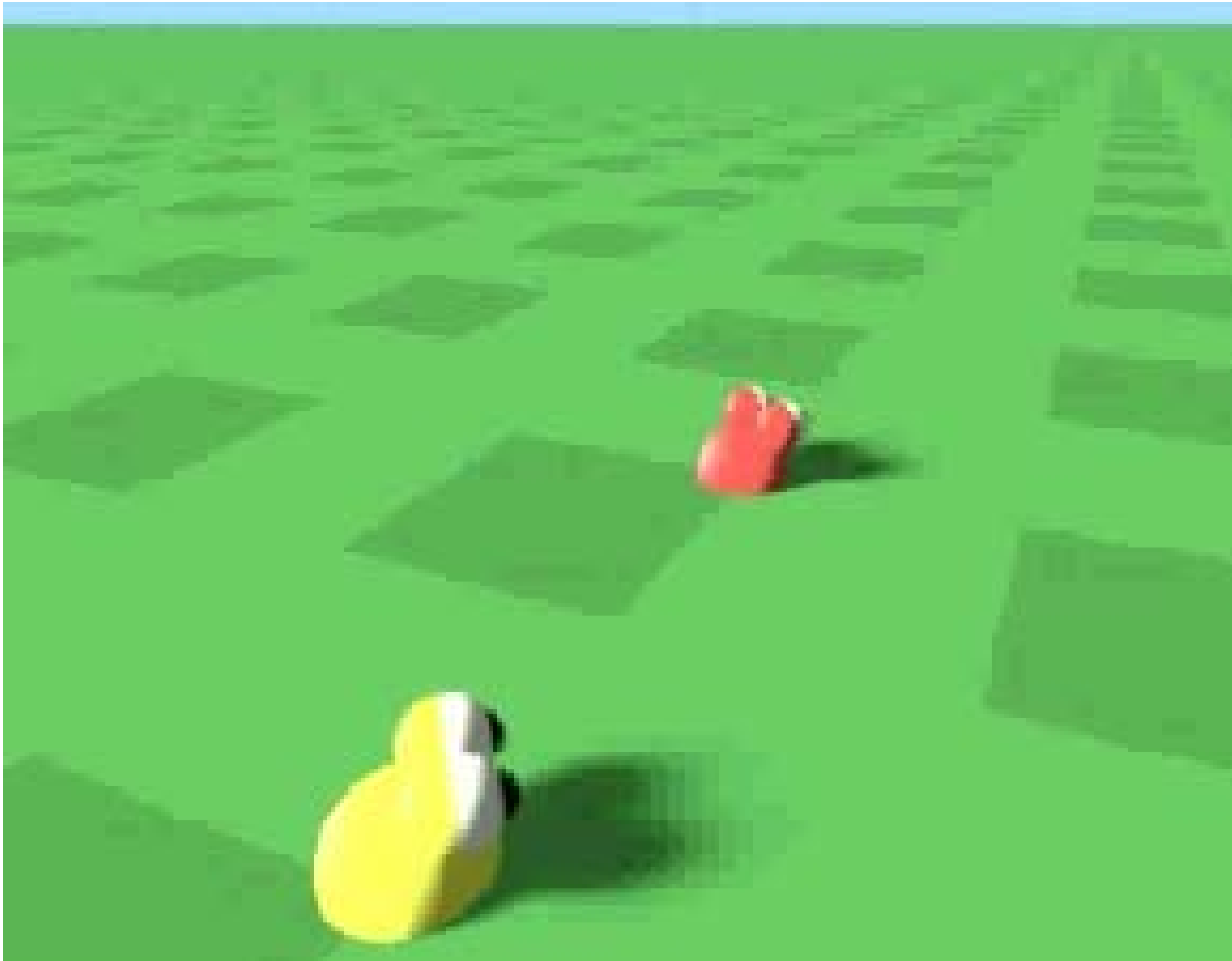
Evaluation

Evaluation

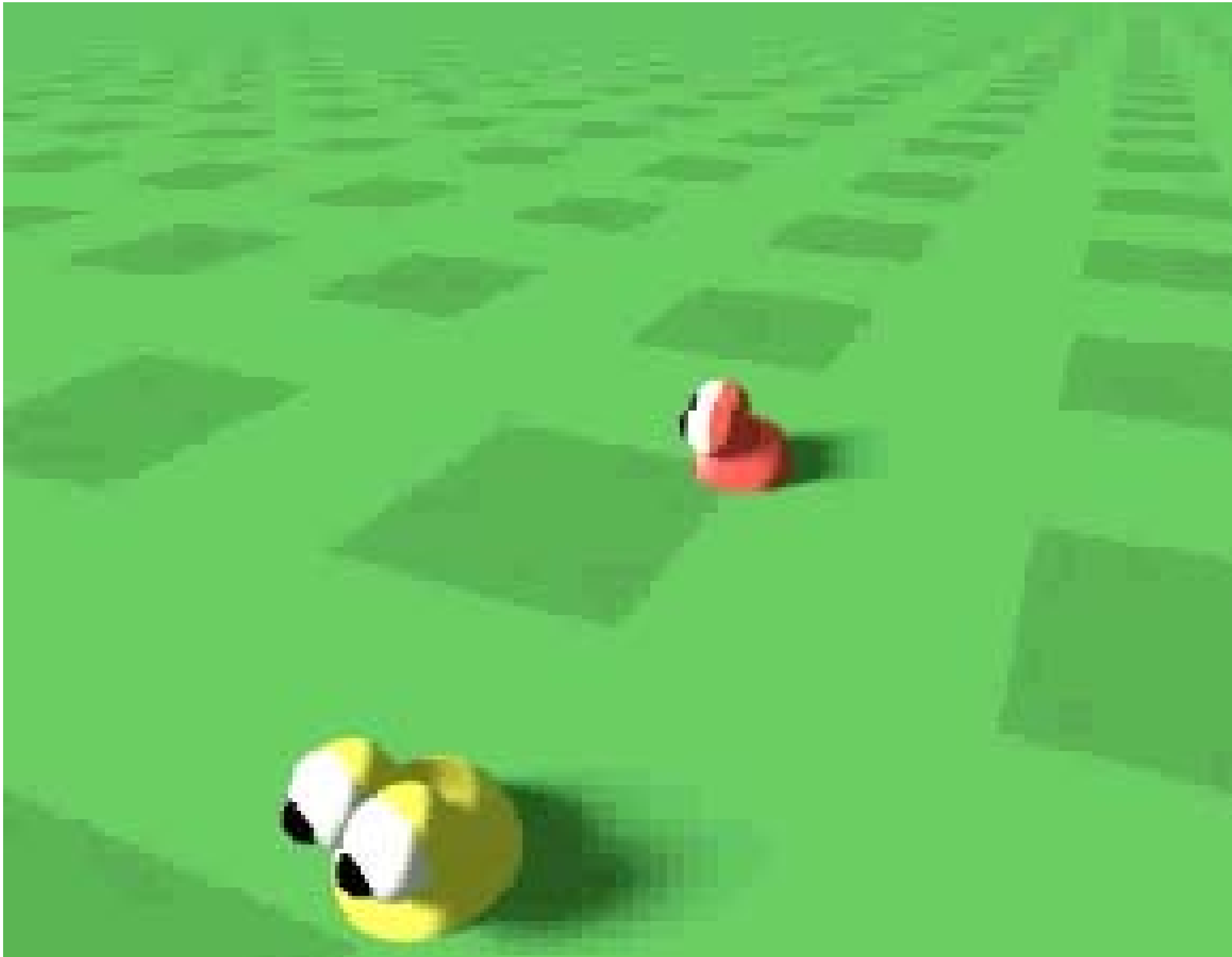
Evaluation:
All in the population of pursuers are tested against the same best-of-last-generation evader.
And vice versa.

Several trials from random starts:
Evader fitness = how long before caught
Pursuer fitness = ++ for 'approaching evader'
+ bonus for hit , sooner the bigger

Random Start – gen 0



A Successful run – gen 999



Potential Circular Trap

That last picture showed successful pursuers/evaders from generation 999

But at gen 0, there was a pursuer which failed to catch an evade
and at gen 999 likewise.

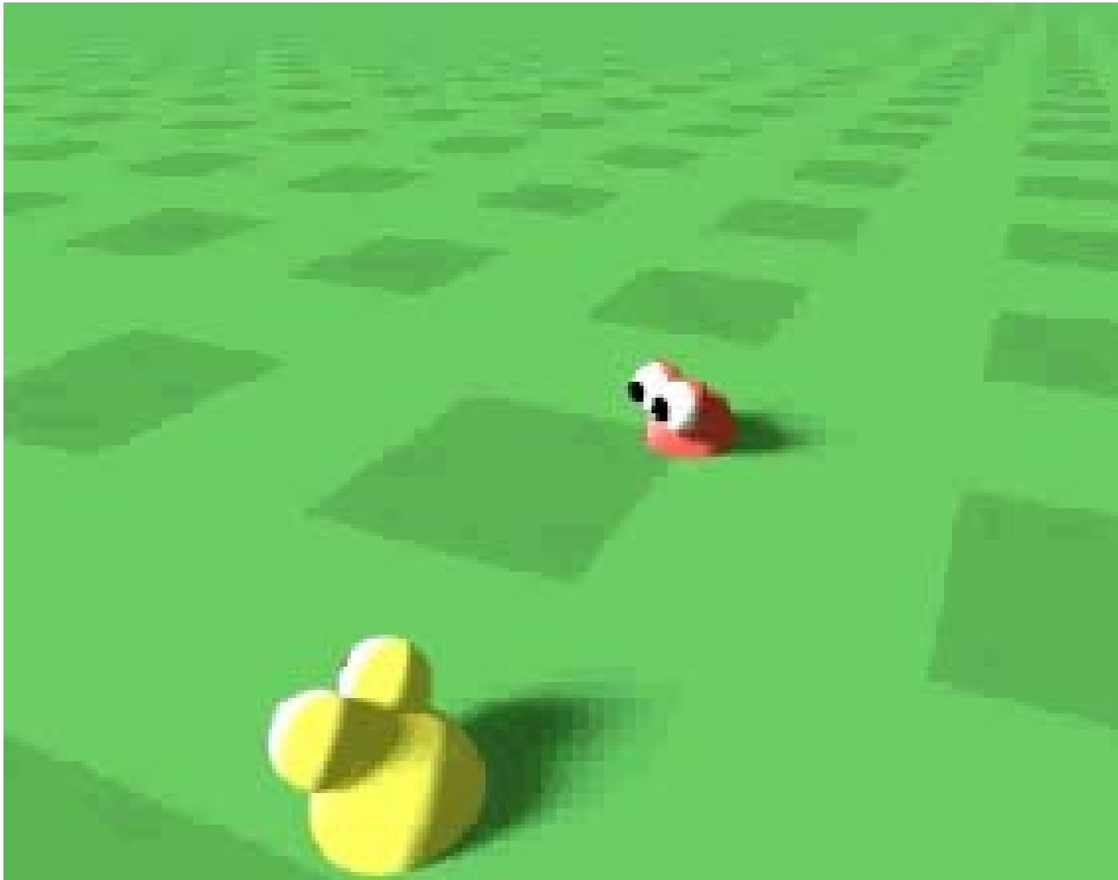
So in what sense has there been any 'advance'?

Possibility of 'no real advance' in coevolution
-- cf Stone Scissors Paper game, no strategy can be supreme fo
ever.

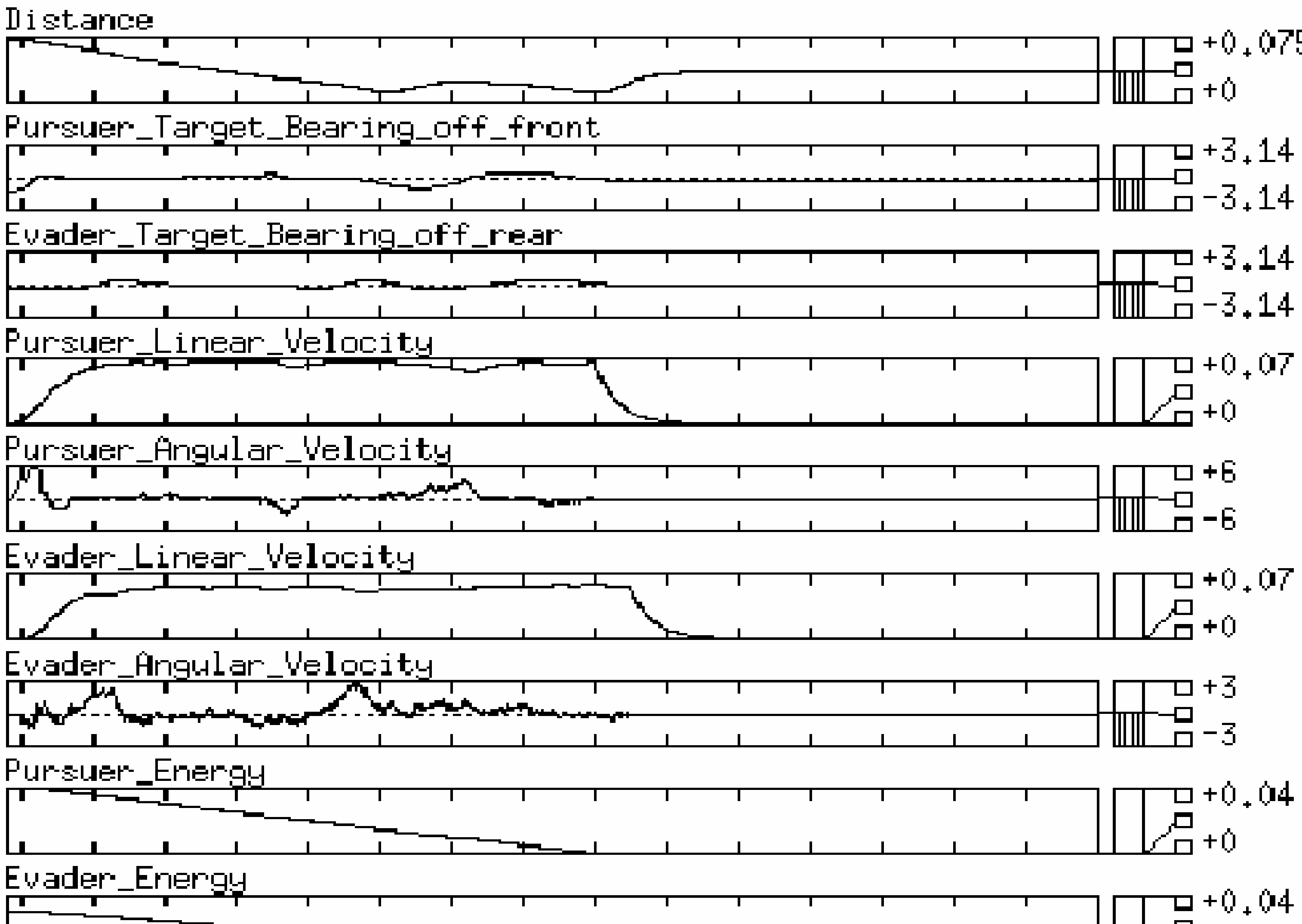
Possible Variations

Test evader from gen 200 against pursuer from gen 999.

Later work extended this idea, of monitoring current gen against best of previous gens.
– Does this escape from the circular trap?



Analysis of behaviour



Applications

Can coevolution be used for engineering purposes?

Here is an example

Coevolving Parasites

"Coevolving Parasites improve Simulated Evolution as an Optimization Procedure". W. Daniel Hillis. In Artificial Life II, Langton Taylor Farmer Rasmussen (eds) Addison-Wesley (1991) pp 313-322

Danny Hillis -- Connection machines --powerful very distributed p machines.

This work done in late 1980s, 64,536 processors, populations 50 1000000, 'about 100 to 1000 generations per minute'

Evolving minimal *sorting networks*

Sorting Networks

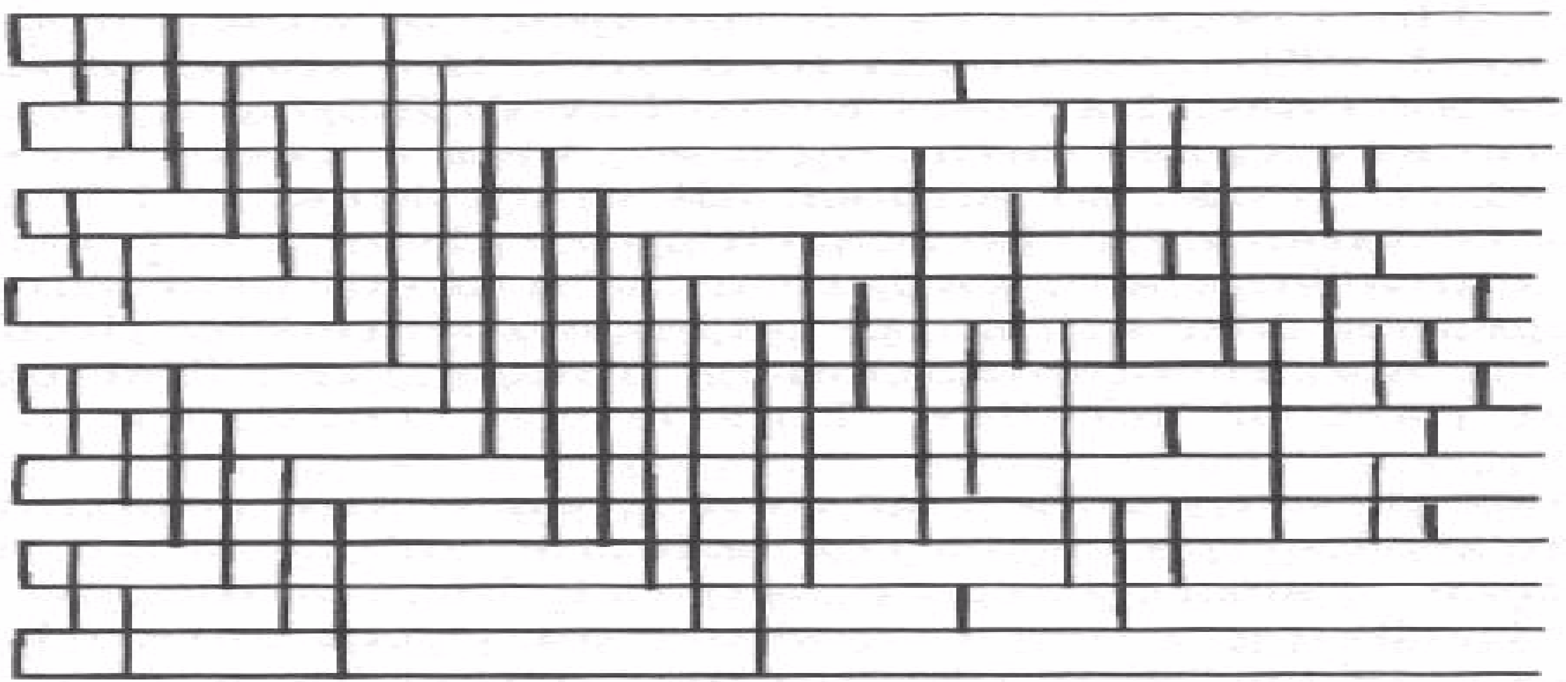
A sorting network is something that can be given **any** scrambled l
N objects with different values (here N=16) --- and it is in effect ar
algorithm that will systematically sort the list into order by a seque
'compare and maybe swap's.

The sorting network is a series of pairs of numbers,
[a b] which can be interpreted as:-

✓Compare the ath and bth items in your scrambled list.

✓ If in wrong order, swap, otherwise leave

Picturing Sorting Networks



Visual way to represent, 16 rows represent the 16 items to be re-ordered. Starting from left, the vertical bars show rows to be compared/swapped.Numbering rows from 0 to 15, above swaps a
[0 1] [2 3] ...[14 15] [0 2] [4 6] [8 10] ......

Minimal Sorting Networks

The previous diagram has a total of 60 swaps and was (in 1991) t
shortest-known, discovered by MW Green

It is a **perfect** sorter, in that if you present it with **any** scrambled l
after going through all the 60 swaps from left to right then the list c
out perfectly ordered.

[note: for swaps shown as bars in same vertical column, it will no
matter which is done first]

The problem is to find the shortest network, ideally better than this
which still sorts anything.

How to check if it works

Do you have to check if it sorts all possible combinations of numb the list – NO!

It can be shown that if a network correctly sorts any scrambled lis and 1s (so that it finishes up with all the 1s at the top, all the 0s at bottom), then the network will also sort any list of real-valued item

So can test a 16-network exhaustively with only 2^{16} tests (about 3 – instead of 16 factorial (about 2×10^{13}).

But this is still a lot of tests -- can one save time? – YES!

Genetic Representation

We need a genetic encoding, so that strings of characters represe
possible sorting networks.

But we are not sure how long any sorting network will be before w
– after all, we are looking for the shortest.

Hillis chose a sort-of-diploid encoding

haploid = 1 string
diploid = 2 strings

Diploid encoding

A codon pair looks like this or this:

```
         --------------                                      --------------
.... .... 0011 0101 ... ... ... ... ... ... ... ... 0011 0101 ...
.... .... 0011 1000 ... ... ... ... ... ... ... ... 0011 0101 ...
```

Where top and bottom are different, on left, this means test/swap [3 5] (binary 0011 and 0101), followed by test/swap [3 8] --- total 2 test/swaps

Where top and bottom are same, as on right, it is just test/swap [3 5] --- only one test/swap

Diploid encoding (ctd)

A full genotype is 60 such codon-pairs,

--- hence encoding between 60 and 120 test/swaps.

cf: homozygous / heterozygous (a bit different !)

Scoring

The population is initialised with everyone having the same first 3
exchanges (that are known to be sensible),
and thereafter randomised.

Then each network is tested on 'how well it sorts --
 the percentage of input test scrambled lists which it sorts correctl

Rather than testing on all 2^{16} test cases, it could be tested on a ra
sample.

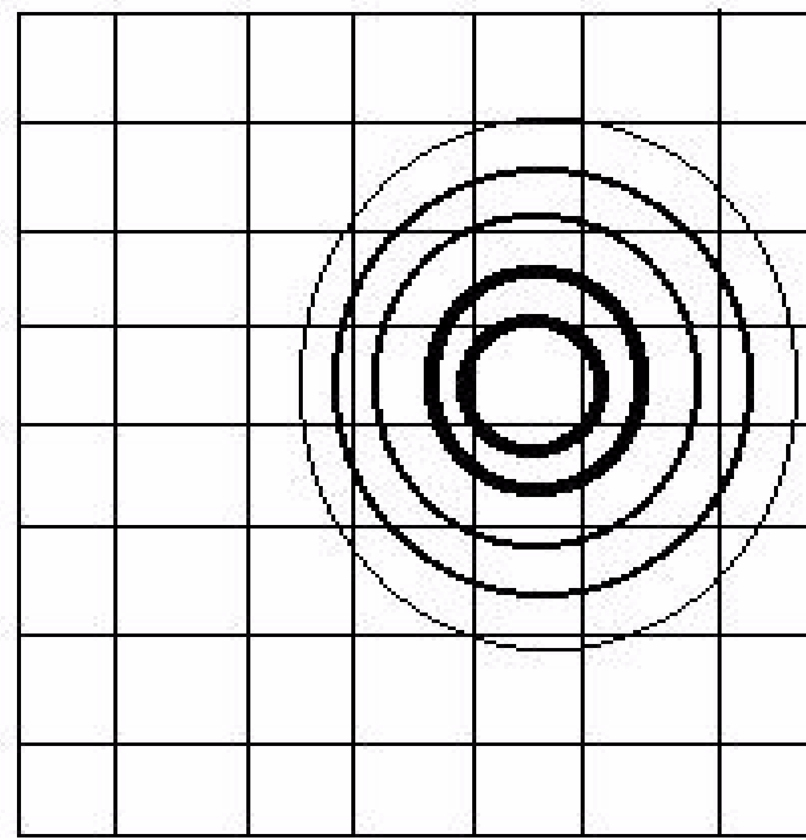
OR (see later) the test cases could be chosen cleverly – coevoluti

Spatially Distributed GA

Tournaments:

pick pairs of contestants in local neighbourhood

(Gaussian spread,
nearer is more likely)



Reproduction

Tournament: from pair of contestants, compare scores, winner over-writes loser (ie then has 2 copies).

Mating: then select mates locally, with same principles

Recombination to produce offspring
(Hillis actually had 15 crossover points '1 per chromosome')

Mutation: one bit-flip per 1000 sites.

Results – without coevolution

Typical run like this -- without coevolution -- for up to 5000 generations, with a popn of 64536.

Best scores = sorting networks of 65 exchanges -- target was 60.

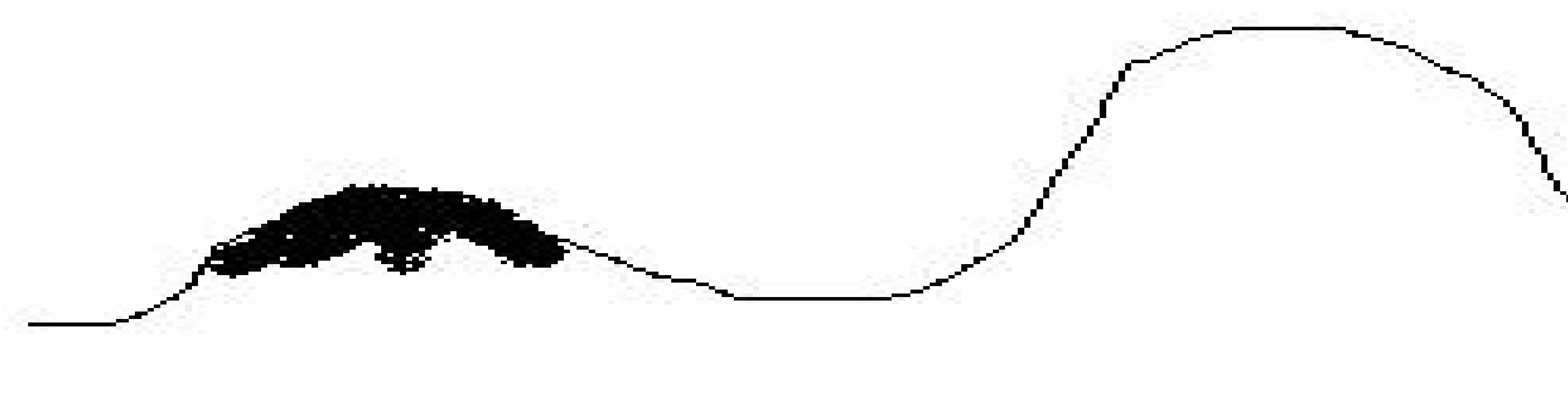
How can one improve this through coevolution ?

Inefficiencies - 1

Inefficiencies - 1

Two main sources of inefficiency in the GA without coevolution:

(1) Local optima -- once the population had found a 3/4 decent so
quite probably all the neighbouring solutions (genetically similar) w
less fit -- so the population would have to cross a valley to reach 'h
ground'.



Inefficiencies - 2

(2) Inefficiency in testing -- once popn was ¾ decent, they all pass
most of the test cases, so little differences in scores.

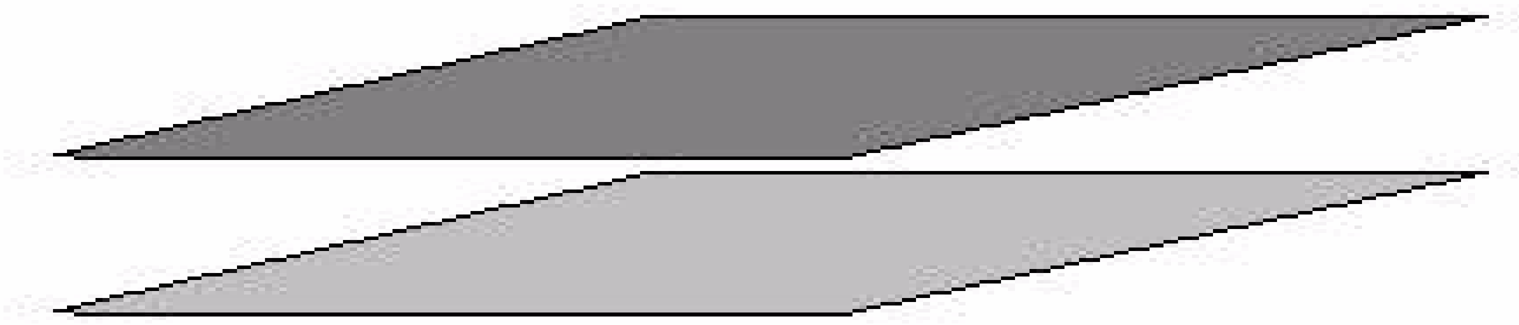
The answer: co-evolve a separate population of parasite test-cas
which themselves have a fitness function designed to make them
as hard as possible for the sorting networks.

This solves both inefficiencies (1) and (2).

Parasite coevolution can generate genetic diversity

(cf. W Hamilton)

2 populations – sorters and parasites



The population of sorting networks is already spatially distributed
grid. Have a population of parasites likewise distributed on a simil
overlaid.

Each parasite is a genetically specified group of 10 to 20 test case
rather than all the 2^{16} possible ones.

Scoring each population

Each sorting network is tested against the parasite that is on corresponding grid square. The score of the sorting network is 'wh proportion of tests does it pass'

The score of the parasite is 'how many tests does it fail the sorter

Networks get selected, mated, and reproduce on their grid, parasi completely separately on theirs.

Results improved to a minimum size of 61
(has it been beaten since?)

Benefits of Coevolution

Prevents getting stuck in local optima -- as soon as this happens
parasites evolve to zap them.

Population is in a constant state of flux.

Second advantage: testing is more efficient --
need only test on a few difficult test cases, which themselves cha
appropriately according to circumstances.

Hence computationally more efficient.