Non Symbolic AI Lecture 12

Some more on Evolutionary Algorithms

1) Genetic Programming – GP
2) Species Adaptation Genetic Algorithms -- SAGA

Genetic Programming

GP (a play on General Purpose) is a development of GAs where the genotypes are pretty explicitly pieces of computer code.

This has been widely promoted by John Koza, in a series of books and videos, eg:

Genetic Programming, John Koza, MIT Press 1992
Followed by volumes II and III

Sort of: using NSAI techniques to evolve programs (of symbols!)

GP - Problems

At first sight there are a number of problems with evolving program code, including

(Problem 1) How can you avoid genetic operators such as recombination and mutation completely screwing up any half-decent programs that might have evolved?

(Problem 2) How can you evaluate a possible program, give it a fitness score?

GP solution to problem 1

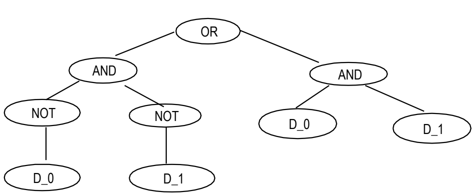
Both these 2 problems were basically cracked in work that predated Koza:

N. Cramer *A representation for the adaptive generation of simple sequential programs* In J Grefenstette (ed) *Proc of Int Conf on Genetic Algorithms*, Lawrence Erlbaum, 1985.

(1) Use LISP-like programs, which can be easily pictured as rooted point-labelled trees with ordered branches.

Picturing a Lisp program

(OR (AND (NOT D_0) (NOT D_1)) (AND D_0 D_1)



Recombining 2 Lisp programs

Picture each of 2 parent Lisp programs in tree form.

Then **recombination** between 2 parents involves taking their 2 trees, choosing random points to chop off sub-trees, and swapping the sub-trees.

This maintains the general form of a program, whilst swapping around the component parts of programs.

Mutating a Lisp program

EASy

Mutation in GP is rarely used:

"Nonetheless, it is important to recognize that the mutation operator is a relatively unimportant secondary operation in the conventional GA", says Koza
(op. cit. p. 105), citing Holland 1975 and Goldberg 1989.

When it is used, it operates by randomly choosing a subtree, and replacing it with a randomly generated new subtree.

GP solution to problem 2

EASy

How do you measure the fitness of a program? (this was the 2nd problem mentioned earlier)

Usually, fitness is measured by considering a fixed number of test-cases where the correct performance of the program is known.

Put a number on the error produced for each test-case, sum up the (absolute) value of these errors => fitness.

Normalising fitness

EASy

Often there is some adjustment or normalisation, eg so that normalised fitness nf ranges within bounds 0 < nf < 1, increases for better individuals, and sum(nf)=1.

Normalised fitness => fitness-proportionate selection

A typical GP run

EASy

- has a large population, size 500 up to 640,000
- runs for 51 generations (random initial + 50 more)
- has 90% crossover -- ie from a population of 500, 450 individuals (225 pairs) are selected (weighted towards fitter) to be parents
- and 10% selected for straight reproduction (copying)
- no mutation
- maximum limit on depths of new trees created
- selection is fitness-proportionate

Different problems tackled by GP

EASy

Koza's GP books detail applications of GP to an enormous variety of problems.

- Eg finding formulas to fit data
- control problems (eg PacMan)
- evolution of subsumption architectures
- evolution of building blocks (ADF automatic definition of (sub-)functions)
- etc etc etc

Choice of primitives

EASy

In each case the primitives, the basic symbols available to go into the programs, must be carefully chosen to be appropriate to the problem.

Eg for PacMan:
Advance-to-Pill
Retreat-from-Pill
Distance-to-Pill
... ...
IFB(C D) If monsters blue, do C, otherwise D
IFLTE(A B C D) If A<=B, do C, otherwise do D

Criticism of GP

GP has been used with some success in a wide range of domains. There are some criticisms:

Much of the work is in choice of primitives

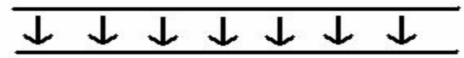
Successes have not been in General Purpose ('GP') programming -- very limited success with partial recursive programs (eg those with a DO_UNTIL command)

But different Evolutionary Algorithm ADATE is far better than GP on this.

Wide and short, not long and thin

GP practitioners such as Koza typically use very large populations such as 640,000 for only 51 generations.

This is closer to random search than to evolution.
Very WIDE and SHORT



SAGA

SAGA, for Species Adaptation Genetic Algorithms, is in many respects the very opposite of GP. Papers on my web page
http://www.informatics.susx.ac.uk/users/inmanh/

It is intended for *very long term evolution*, for design problems which inevitably take many many generations, quite possibly through incremental stepping-stones.

Long and thin, not wide and short

So in contrast to GP, there is typically a relatively small population (30-100) for many generations (eg 1000s or 10,000s).

'*Long and narrow*' not 'wide and shallow'

The population is very largely 'genetically converged' -- ie all members genetically very similar, like a plant or animal ***SPECIES.***

Mutation vs. Recombination

Mutation is the main genetic operator, adding diversity and change to the population.

Recombination is only secondary (though useful).

This is completely contrary to the usual emphasis in GP

Convergence (1)

The term 'convergence' is often used in a confused way in the GA/GP literature. People fail to realise that it can be applied with at least two different meanings.

(1) Genetic convergence -- when the genetic 'spread' of the population has settled down to its 'normal' value, which is some balance between:

- ❑selection of the fittest -> reduces spread, and
- ❑mutation -> increases spread.

Convergence (2)

EASy

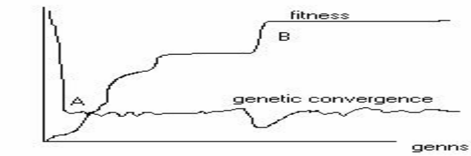
(2) convergence of the fitness of the population onto its final value, or (very similarly) convergence of the 'search' of the popn onto its final resting-place.

In sense (2) people talk of 'premature convergence', particularly when they are worried about the population converging onto a local optimum in the fitness landscape, one that is very different from the global optimum.

A common, completely false, myth is that convergence in sense (1) implies convergence in sense (2).

Monitoring Genetic Convergence

EASy



B is the point of convergence (defn 2), often after 'punctuated equilibria'

A is the point of genetic convergence (defn 1), which may well be (surprisingly?) within the first 10 or so generations !

Long term incremental evolution

EASy

SAGA was originally developed with a view to long term incremental evolution, where one would start with (relatively) short genotypes encoding (eg) relatively simple robot control systems ...

... then over time evolution would move to longer genotypes for more complex control systems.

In this long term evolution it is clear that the population will be genetically converged for all bar the very start

… also long term non-incremental evolution

EASy

BUT though SAGA was originally developed with the view of long term incremental evolution (where genotype lengths probably increased from short, originally, to long and then even longer…)

It soon became apparent that the lessons of evolution-with-a-genetically-converged species were **ALSO** applicable to *any* long term evolution, even if genotype lengths remain the same!

Consequences of convergence (1)

EASy

It soon became apparent that even with fixed-length genotypes, one still has genetic convergence of the population from virtually the start -- even though this is not widely recognised.

Consequences: recombination does not 'mix-and-match' building blocks quite as expected -- because typically the bits swapped from mum are very similar to the equivalent from dad.

Consequences of convergence (2)

EASy

Evolution does not stop with a genetically-converged population (GCP) -- eg the human species, and our ancestors, have evolved as a GCP (… or *species)* for 4 billion years, with incredible changes.

Mutation is the main engine of evolutionary change, and should be set at an appropriate rate – which to a first approximation (depending on selection) is:

… …

Junk DNA

EASy

A high proportion – indeed most – of human and other animal/plant DNA appears to be ‘junk’.

I.e., not used, not ‘translated’ or ‘interpreted’. So what is it doing there?

“Rubbish is what you chuck out, junk is what you put in the attic in case you might need it later”

Some genes might be unnecessarily duplicated by accident – and later the spare copies mutated into something else useful.

Optimal mutation rate – for *Binary-type* genotypes

EASy

One proposal (with some small print):

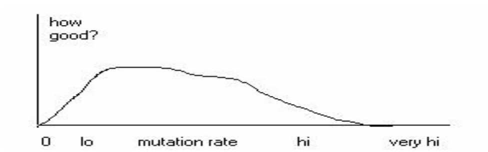
1 mutation in the expressed (non-junk) part of the genotype

CF phages with 4500 'characters' in genotype
humans with 3,000,000,000 'characters‘

-- both have something of the order of 1 mutation per genotype (or at least per non-junk part)

Why should there be an optimal mut-rate?

EASy



Mutation rate too low, in limit zero, would mean no further change, evolution ceases -> no good

Mutation rate too high, eg every bit flipped at random, implies random search -> no good.

What decides the ideal rate?

EASy

Some ideal rate (or range of rates) inbetween -- but where ?

Very rough version of argument: to a first approximation (tho not a 2nd !) at any stage in evolution some N bits of the genotype are crucial -- any mutations there are probably fatal -- while the rest is junk.

The *error threshold* shows that maximum mutation rate survivable under these circumstances is of the order of 1 mutation in the N bits

So what is SAGA?

EASy

Species Adaptation Genetic Algorithms

SAGA is not a specific GA, it is just a set of guidelines for setting the parameters of your GA when used on any long term evolutionary problem -- with or without change in genotype length

1) Expect the population to genetically converge within the very first few generations

2) Mutation is the important genetic operator

… SAGA ctd

EASy

3) Set the mutation rate to something of the order of 1 mutation per genotype. Subject to small print below.
4) Small print 1: it is the mutations in the non-junk part that matter – so if only 1/3 is non-junk, you will need 3 mutns per genotype,
5) Small print 2: this is only for standard selection pressure (as in tournaments of Microbial GA). Stonger selection needs more mutation. If abnormal selection pressures, for 1 substitute log(S) where S is the expected no. of offspring of the fittest member

… SAGA ctd …

EASy

Often there is quite a safe range of mutation rates around this value -- ie although it is important to be in the right ballpark, exact value not too critical

Recombination generally assists evolution a bit

Expect fitness to carry on increasing for many many generations

Mutation rates – Summing up

EASy

IMPORTANT: very different rules for binary and for real-valued genotypes.

BINARY or similar: Here a mutation flips from 0 to 1, or from 1 to 0.

Mutation rate – use SAGA principle of 1 mutation per genotype, adjusted if necessary for junk DNA and for abnormal selection pressure.

This may not be the best rate – but at least a good starting place.

Mutation Rates – for real-valued genotypes

EASy

BUT the rule for real-valued genotypes is different.

Real-values such as when a genotype encodes the weights of an Artificial Neural Network.

Mutating a real number (float or double) – you are probably best off changing it by a small "creep factor" (cf Lec 3).

And then it may well be a good idea to change *every* locus on the genotype by a small creep factor – as compared to changing just *one* locus in a binary genotype by a big change (0 / 1)

Lots of little changes

EASy

In, e.g. an Artificial Neural Network, a learning algorithm typically jiggles *every* weight by a little bit.

It is much the same when, in a GA with real-values on the genotype, a mutation jiggles (or 'creeps') *every* locus, every real value, by a little bit.

Selection pressures

EASy

It is very tempting to use a high selection pressure for Genetic Algorithms/Evolutionary Algorithms --

~~"Just select the very fittest as a parent and throw away the rest"~~

This is almost always a bad idea – evolutionary search gets stuck in a dead end.

I recommend standard selection – as in Microbial GA, pick winner of tournament of size 2 – and then adjust the mutation rate appropriately. But you can experiment.

The End

EASy