LUND UNIVERSITY

Genetic Algorithms

Jacek Malec
AI@CS

email: jacek.malec@cs.lth.se
March 26, 2009

AI@CS
Department of Computing Science

LUND UNIVERSITY

Plan for today

- What is a genetic algorithm?
- Degrees of freedom.
- Some examples.
- Co-evolution, SAGA, Genetic Programming, Evolutionary Strategies, ...

These slides have been prepared with help of Inman Harvey's lecture notes on this topic.

AI@CS
Department of Computing Science

LUND UNIVERSITY

Randomised search

- Hypothesis space;
- Search for the optimal solution (hypothesis);
- Evaluation function;
- How to decide that we are happy?

The running example: find the (an) optimal control system for a simple robot (the Braitenberg vehicle).
Maybe: find best parameters for the walking gait of an AIBO.
Find the rules (a decision tree) for a simulated soccer player.
...

AI@CS
Department of Computing Science

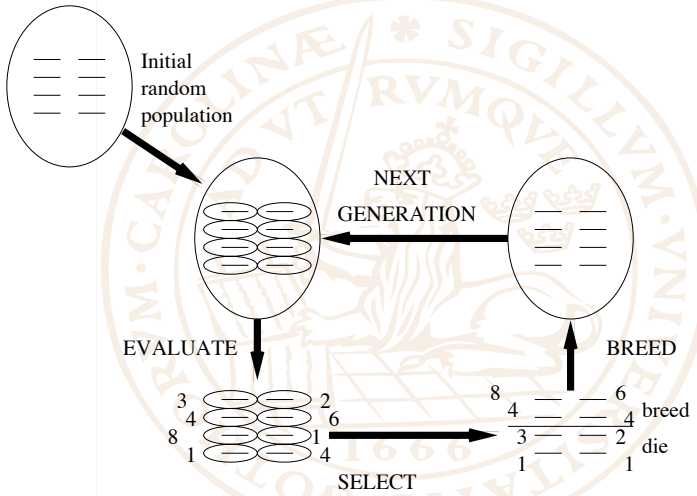
LUND UNIVERSITY

Terminology:

- individual - object with a particular genetic structure
- chromosome - genetic structure represented as a string of 1's and 0's
- allele - a bit in a string representing a chromosome
- genotype - the genetic string associated with an individual
- fitness - value of an individual determined by the evaluation function
- generation - a set of individuals
- crossover site - an index into a bit string

AI@CS
Department of Computing Science

The GA cycle



Decoding genotypes

In our case a genotype encodes a robot control system or its part (sometimes with the perception part too).
Genotypes can be binary strings, real numbers, characters, program segments, ...
This decoding part is 99% of the code in a GA program, and is domain-specific.
The GA cycle is 1% of the code. But is critically important!
In an evolutionary run, almost all the time you evaluate thousands of robots. If the GA can be set up so as to require fewer evaluations, you can reduce the total time required to a small fraction.

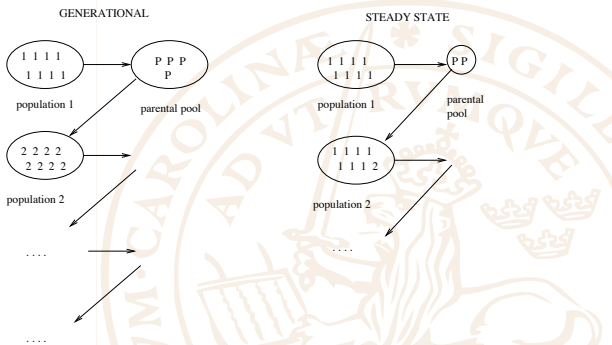
How can you vary a GA?

- Population size
- Generational or Steady State
- Selection
- Recombination
- Mutation
- Saving evaluations
- Distributed GA
- Incremental evolution
- Choosing a good genotype - robot encoding

Population size

If it is too small, then it ultimately reduces to something like hill climbing by random search about a single point.
GA give you much more than this: their power lie in *population search*.
If it is too big, then it takes too long to evaluate the whole population before you get to the next generation.
How big is best?
Theory is obscure and problem dependent.
Heuristic: 30 to 100 unless you have a special reason otherwise.

Generational or Steady State



Which is best? Not so important, but
Steady state allows distributed independent evaluation of many individuals, without having to maintain global coordination. Also convenient for tournament selection.

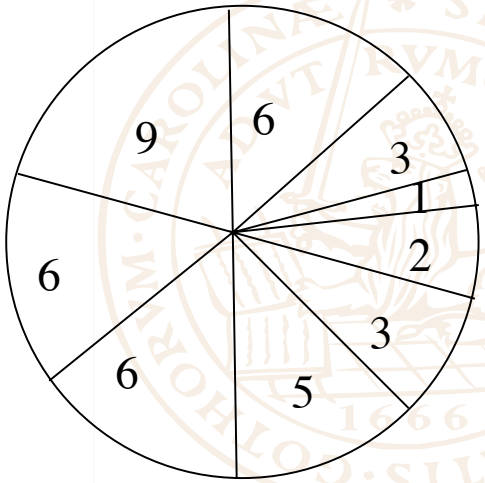
Selection

Each individual (robot, player) is given a score, e.g., in the range 0 to 100. Higher scores give higher probability of being a parent. How do you map scores to reproduction probability?

Beginner's way: roulette wheel selection (apple pie) — fitness-proportionate selection.

Add up all the scores and give each member of the population a probability in proportion to its share of the total fitness.

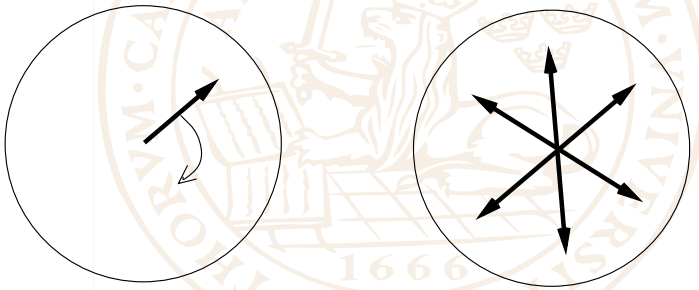
Selection



Variance in selection

To select a parent you can

1. apply one pointer n times;
2. apply an n-pointer once.



Variance cont.

With (1), if an individual has expectation of 1.4 offspring, it can actually have 0, 1, 2, ..., even n offspring.

With (2), it will have either 1 or 2 offspring - 60% chance of 1, 40% chance of 2.

Less variance in number of actual offspring is probably good.

Problem with

fitness-proportionate selection:

If at early stages most individuals score 0/100 and just one scores 1/100, then *all* the population will descend from this one. This is too large selective pressure.

If at later stages all the individuals score between 80/100 and 85/100, then they will all have very similar number of offspring. Too little selective pressure for continued improvement.

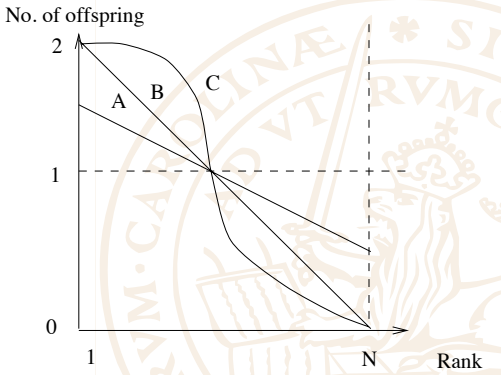
Rank-based selection

This avoids the problems of fitness-proportionate selection and keeps the selective pressure constant.

Rank the population according to fitness and allocate the probability of having offspring *solely* on the basis of their ranking.

Typically linear, with top-rank expected twice the average, middle-rank average number of offspring and bottom-rank zero offspring. Can be linear with different slopes or nonlinear.

Rank-based selection



A: linear from 1.5 to 0.5

B: linear from 2.0 to 0.0

C: nonlinear

Truncation selection

One version of rank-based selection: the bottom $x\%$ die with no offspring, which exclusively come from the fitter group.

E.g. the bottom 80% have no offspring while the top 20% produce enough offspring to replace the whole population, 5 each.

Warning: by conventional GA standards, this is much too high a selective pressure. But if you have a special reason, break the rules.

Exploration vs. exploitation

It's important to keep balance between

- **exploitation**: produce copies of what you know is good
  versus
- **exploration**: try all sorts of variants.

Too high selection leads to overbalance towards exploitation.

Tournament selection

This is convenient for a steady-state GA and is equivalent to a linear rank selection.

- Pick 2 members at random, compare their scores, pick the winner as a parent.
- Repeat for the second parent.
- Produce one child.
- Kill somebody and replace with the child.

Who to kill?

- A random member of population — $\sigma = e$, or
- The *loser* of another tournament — $\sigma = e^2$.

In normal linear ranking, generational — $\sigma = 2$

Selection pressure

σ is the *selection pressure* (selection quotient):

ratio of the best individual's selection probability to the average selection probability of all indiviuals in the selection pool.

Elitism

Many standard GAs keep an unchanged copy of the best member of each generation for the next generation.
An alternative tournament selection, with automatic elitism (with $\sigma = e$) is:

- Choose parent at random, without regard to fitness or ranking!
- Choose member to die as *loser* of another tournament.

But in real world evaluations are always noisy and unreliable, so no form of elitism is guaranteed to retain the best individual.

Rank-space selection

Intended for promoting exploration.

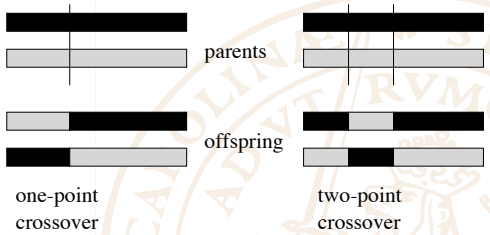
- Create a ranking list based on fitness (as previously);
- Create a ranking list based on distance to the best individual;
- Combine the two (sum, whatever you like);
- Create the final ranking list;
- Choose parents on this basis.

Recombination

Recombination in GAs is conventionally thought of as a powerful "mix and match" operator. The recommended orthodox rate: 60% recombination, 40% asexual.

If a compact little section of the genotype codes for some particular behavioural trait, then recombination probably will not break up this trait.

Recombination, cont.



In one point crossover the two ends are always separated by recombination.

Two-point crossover avoids this — generally a good idea.

Mutation

Conventionally a background operator, say 1 in 100 or 1 in 1000 bits mutated.

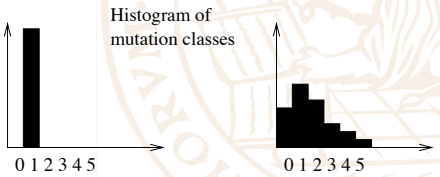
Sometimes not used at all (e.g., in genetic programming).

In incremental evolution (e.g., SAGA) with a genetically converged population, mutation is more important that recombination, as the only source of innovation.

Heuristic: between 1 and 5 mutations per genotype (depends on several factors including selective pressure, amount of junk DNA, etc.)

Mutation, cont.

Apply stochastically at each bit,
e.g. if using expected number 1 mutation per genotype, which has a length 100, actually apply with 1/100 probability at each locus.
This way you get a Poisson distribution: some genotypes get 0 mutations, some get 1, 2, 3, ...



Exactly 1 mutation per genotype

Expectation of 1 mutation applied independently in each locus

Saving evaluations

Almost all of the time a GA is doing the evaluations.

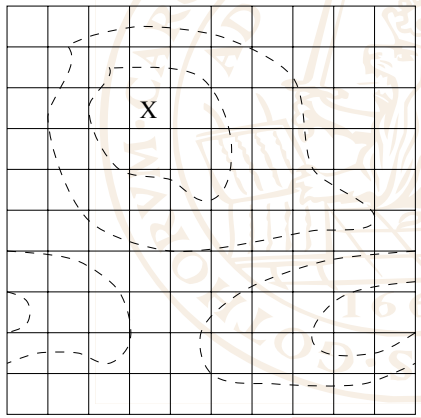
When the population is genetically converged (genetically very similar), often an offspring will be identical to a member of a previous generation.

If the evaluation is deterministic, it is very efficient to save all evaluations and check before each new evaluation whether the value has already been recorded (hash tables are useful).

But robot evaluations will be noisy, so in this case you might want to resample for a new evaluation.

Distributed GAs

Members of the population are thought of as distributed, one per square, on a toroidal grid (e.g. 10×10).



Distributed GAs, cont.

All selection, recombination and replacement operations on any individual (say X) are limited to a local area around X (e.g. 3×3, or a Gaussian centered on X).

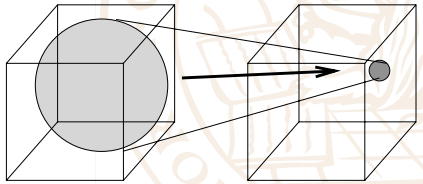
These local areas give effectively a small population allowing drift, and different sub-populations exploring different optima
but
since these areas overlap, any significant improvement can spread through whole population.

GA users of this method often strongly recommend it.

Incremental evolution

Normally GAs are used for *optimisation*.

A single well-defined problem, of fixed dimensionality gives fixed genotype length coding for potential "solutions".



initial random population spans the whole space

final converged population

Incremental evolution, cont.

But natural evolution is nothing much like this!

The problem is often changing (cf. coevolution).

Some (not all) evolutionary pathways lead from "simple" organisms in simple environments to complex organisms capable of dealing with more complex tasks.

We want this incremental process in evolutionary robotics.

SAGA

Species Adaptation Genetic Algorithms

Sussex group: Cliff, Harvey, Husbands

With progress from *simple robots for simple tasks* $\rightarrow$ *complex robots for complex tasks* is assocated genotype length evolution *short* $\rightarrow$ *long*.

A genetically converged species searches through a genotype space of increasing dimensionality!

GAs need to be adjusted for this.

SAGA, cont.

- In SAGA mutation is much more important than recombination.
- When genotype length is allowed to increase, this should happen *gradually*.
- The selective pressure σ should be kept constant — rank or tournament selection gives this.

A note

The most crucial part of evolution is the decision on how the genotype is going to encode the robot's control system (perhaps also sensory or physical morphology).
The nature of this helps to determine the nature of the fitness landscape on which evolution operates.
Smooth, rolling fitness landscapes are much better than jagged, rugged ones.

Hence the need to worry about morphogenesis and noise....

Theorems

Schema theorem: interesting patterns survive evolution;

No Free Lunch theorem: there exist no optimal parameter settings for all possible fitness functions

Convergence, limit theorems, Markov chains, ...

More?

- Genetic Programming;
- Classifier Systems;
- Evolutionary Algorithms;
- Evolutionary Strategies;
- Artificial Life;
- ...