Genetic Algorithms and Genetic Programming
Lecture 11

Gillian Hayes

3rd November 2006

School of informatics

Lectures 11 and 12 : Designing a GA

- When should a GA be used?
- What to represent and how to represent it
  - Encoding the candidates
- The mechanics
  - Evaluating the candidates
  - Selection of the fittest
  - Crossover operators
  - Mutation operators
  - Population models
- The parameters
  - Setting the parameters

- Evaluating the system
  - Did it work? How do we know?
  - How many experiments should we do?
- Summing up

When should a GA be used?

- Large or very large search space
  Noughts and crosses vs. protein folding
- A sufficiently good solution is good enough
  Exam timetabling
- Fitness landscape is not smooth and unimodal
  Optimal headphone loudness vs. setting value on a mixing desk
- Fitness landscape is poorly understood
  Find Flatiron building in Manhattan vs. Paris Left Bank bistro
- Fitness function is noisy and/or complex
  Sensory input or performance in noisy/unpredictable world

- No good algorithms exist to solve the problem

  Timetabling?

- Good local search operators exist

  Building a plan

- The problem is weakly compositional

  TSP vs. Lottery Extra

♠ Linux kernel tuning using a GA (Moilanen): chromosome is string of Linux kernel internal settings, fitness function is performance under some workload (benchmark workloads)

♠ TSP, knapsack, bin-packing, design of concert-hall acoustics, (simulated) F1 cars

Representation: Encoding the candidates 1

What shall we represent?

- The knapsack problem
- Exam timetabling
- Layout of plants and trees in a plot in JCMB

Representation: Encoding the candidates 2

How shall we represent it?

- Fixed-length linear binary encodings

  Unnatural. Unnatural orderings. Hamming cliff. Gray codes?
  Theory exists

- Fixed-length linear non-binary encodings

  Real values or characters. NN weights or grammars

- Variable length linear non-binary encodings

  Plans, Prisoner's Dilemma

- Tree-based chromosomes

  GP. Open-ended search space. But unwieldy trees, much junk

Intuition: encode solution in the most natural way possible, then create genetic operators to make it work.

Mechanics: Evaluating the Candidates 1

- Need
  - A set of configurations C – the chromosomes
  - A fitness function $f : C \rightarrow \Re$
  - An additional geometrical/topological/algebraic structure N on C that allows us to define which chromosomes are neighbours – i.e. what says that chromosome A should be arrayed next to chromosome B on our picture of the fitness landscape? How similar are two chromosomes? (Stadler: landscape theory)
- Single candidate fitness function, $f(c_i)$
  - The more fine-grained, the better
  - Should push towards better solutions
- Fitness function, neighbourhood structure, operators all interact

Mechanics: Evaluating the Candidates 2

- We might use fitness sharing for multiple solutions – prevent premature convergence
  - Fitness = Raw fitness/(Some measure of how many others are similar)
  - Reward difference. Speciation. Explore several local maxima

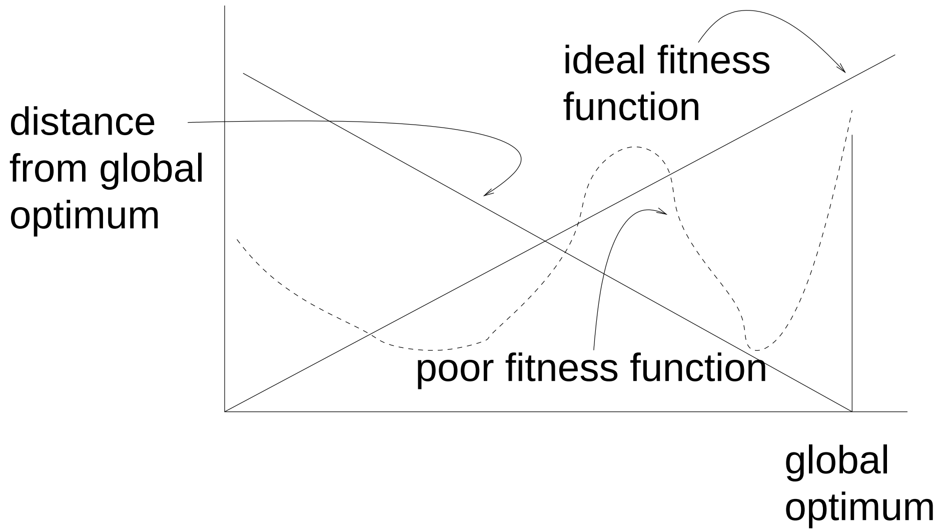
- Round Robin competitions for strategies

- Decode genotype into phenotype and evaluate that

Mechanics: Deceptive Fitness Functions

Fitness function: estimate of how far it is to the global optimum.

What if our estimate is not so good?



Mechanics: Fitness Distance Correlation

Assume you have a set of fitnesses $F = f1, f2, \ldots$ and a set of known distances to the global optimum $D = d1, d2, \ldots$.

$$\mathrm{FDC} = \frac{C}{SF \times SD}$$

SF and SD are the standard deviation of F and D respectively and

$$C = \frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})$$

where $\bar{f}$ and $\bar{d}$ are the means of F and D. C is the covariance of F and D.

Ideally, FDC = -1. (Why?)

Maximally **deceptive** fitness functions: FDC = 1.

Mechanics: Selection Method

Aim: choose parents. Emphasise fitter ones. Balance exploitation and exploration.

- Fitness-proportionate selection
  Often premature convergence

- Rank-based selection
  Fitness(i) $=$ Min $+$ (Max - Min)(Rank(i) - 1)/(N-1) then do FPS
  Max and Min are chosen by you.
  Can also do exponential scaling.
  Preserves diversity, slows selection pressure

- Tournament selection
  Select k individuals. Fittest m go into intermediate population (perhaps

with some probability)

Less computationally expensive (don't evaluate all chroms.)

- Uniform selection

  Lowest/highest fitness in current generation is Min, Max. Select a fitness f uniformly in [Min, Max]. Individual with closest fitness to f is chosen. Maintains genetic diversity – we only want **one** solution of maximal fitness

- Elitism

  Copy some number of fittest individuals into intermediate or next-generation population

  Don't lose good solutions when we've found them until we find better solutions

- and others, e.g. combinations of the above

Mechanics: Selection Method Considerations

- Selection pressure – avoid premature convergence, maintain diversity, exploration vs. exploitation

  How would we detect premature convergence?

- Takeover time – till best individual replaces all others

  Is the best individual good enough?

Mechanics: Crossover

- Single-point, Two-point
- Uniform: choose each child gene with probability p from parent 1
- Try to preserve building blocks (but avoid hitch-hiking)

Attempts to make crossover less disruptive:

– Brood crossover: 2 parents produce several offspring, fittest 2 chosen

– Elite crossover: put offspring into pool with parents, select fittest 2

– Intelligent crossover: crossover hotspots – a template for crossover points that is also evolved

Mechanics: Mutation Operator

- Original aim: to preserve diversity
- Can end up solving the problem
- Allele (point) mutation
- Reordering mutations – inversion or cycling
- Swap mutations – swap 2 random positions
- Intelligent mutations
  - encode p_m onto chromosome and allow GA to evolve it
  - use a schedule to change p_m as a function of time
  - or as a function of fitness
- Choose to suit the problem [To be continued...]