Genetic Algorithms and Genetic Programming
Lecture 12

Gillian Hayes

7th 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

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. See Mitchell Sect. 5.4.

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
- Try to preserve building blocks (but avoid hitch-hiking)
- Uniform: choose each child gene with probability p from parent 1, else 2 – so no linkage between genes. Often p = 0.5 or a bit higher.

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 $p \sim 1/n$
- 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

Mechanics: Population Model and Elitism

- What do we do with the new candidates?

- Generational model: place children from generation n into generation $n+1$, continue until generation $n+1$ is full

- Generational model with elitism: keep some proportion of the best candidates from generation n and give them a free ride into generation $n+1$

- Steady state model (Whitley): newly created children are inserted into the $current$ generation and replace the worst candidates

- Can lead to very high selection pressure (why?)

Mechanics: Spatial Separation

- Island model: evolve independent populations, sometimes allowing a good candidate to migrate across islands
- Simple island model:
  - P independent populations
  - every M generations select one (of best) candidate from one population
  - insert it into all other populations (replace the worst)
- Advantages: encourages diversity, can run in parallel
- Can use different fitness functions on the islands – multicriterion optimisation
- Cellular model: candidates are arranged on a Grid and can only mate with nearby candidates

Mechanics: Maintaining Diversity

- Restrict which individuals can mate with each other, e.g. must be sufficiently different.
- New offspring replace those most similar to themselves
- Fitness sharing (see earlier slide)
- Mate selection – keep a label on each chromosome. It can only mate with chromosomes with same label. Evolve labels
- Island and cellular models
- High p_m

Setting Parameters

- Usual approach:
  - Try a wide range of parameters
  - Using the best, alter one at a time
  - Continue until no improvement possible
- but the parameters interact non-linearly
- Systematic approach: use a lot of CPU time
- Self-adaptation? Fitness of operators based on how many highly fit individuals they contribute to producing
- Starting point: Small population 20–50, crossover rate 0.75–0.95, mutation rate 0.005–0.01 per bit

Evaluating the System

- No. generations vs. no. evaluations
- Cpu time vs. real time (fitness function takes most time)
- Benchmarks: a range of real problems
- Compare with alternative algorithms
- Time to reach solution
- Quality of solution

- $Real$ fitness of solution (to your actual purpose)
- Acceptability to users
- When is one parameter set better than another? Statistics, many experiments with each parameter set

What do Genetic Algorithms Offer?

- Robust problem-solving ability
- Search, optimisation, machine learning
- Good performance on dynamic problems (e.g. job-shop scheduling)
- Ease of implementation
- Hybridisation with other methods
- Anytime problem solving behaviour
- Easy to run on parallel machines

- A competitive solution for many hard problems

Reading: Mitchell Chapter 5 – all of it

Section 10 of Whitley tutorial (notes Chapter 2)