Genetic Algorithms

Alan Dix
dixa@comp.lancs.ac.uk

Manolis Sifalakis
mjs@comp.lancs.ac.uk

CSc355

Lecture Overview

• Evolutionary Computation (EC) and Genetic Algorithms (GA)
• A Brief History
• Terminology from Biology
• Computational Model
• Search space and fitness landscapes (an example)
• GAs versus other search methods
• Why evolution
• Applications of GAs
• Sample Application: Genetic Programming (GP)
• Performance Tuning of GAs
• Reference material (for GAs and GP)

Evolutionary Computation

• Computational systems that use natural evolution (universal Darwinism) as an optimisation mechanism for solving engineering problems

Initial Population of Candidate Solutions  Evolve/Optimise (Natural Selection)  New Population of fitted Solutions

A Brief History

Evolutionary Computation

Evolutionary Strategies

Evolutionary Programming

Genetic Algorithms

• Evolution strategies: Real value parameter optimisation for device models [Rechenberg 1965, Schwefel 1975]
• Evolutionary programming: Evolvable state-transition diagrams (FSM) to produce fit solutions for specific tasks [Fogel, Owens, Walsh 1966]
• Genetic Algorithms: Abstraction and formalisation of natural-adaptation mechanisms for general purpose computations [Holland 1962] ... as opposed to problem-specific algorithm development
• Other independent efforts for evolution-inspired algorithms

Biological Systems: A rough guide

• Living organisms consist of cells
• Each cell contains one or more chromosomes (DNAs & RNAs)
• A set of chromosomes provide the organism blueprint
• A chromosome is divided conceptually in genes (functional blocks of DNA)
• A chromosome is divided conceptually in genes (functional blocks of DNA)
• A (set of) gene(s) encodes a protein - a trait (e.g. eye color)
• Alleles are the possible encodings of a gene (blue, green, red, yellow)
• Locus is the position of a gene in the chromosome

• Genome: Complete collection of chromosomes (genetic material)
• Genotype is a particular set of genes (encoded in chromosomes) in the genome that represent the genetic material of an individual
• Phenotype are the physical or mental characteristics related to a genotype (eye color, intelligence, height, hair type, etc) of an individual
Biological Systems: A rough guide

- Organisms whose chromosomes appear in pairs (most sexually reproducing species) are called diploid, if not they are called haploid.
- During sexual reproduction, genetic recombination (crossover) occurs whereby chromosomes exchange sets of genes to produce a gamete (haploid).
- Mutation is the product of copying errors in the recombination process (biochem action, ext radiation, etc).
- Genetic fitness refers to the probability that a new organism will survive to reproduce (viability) or the number of offspring an organism has (fertility).

Computational Model

- A chromosome is a string representation of a candidate solution to a problem.
- The genes are single digits or subsets of digits at specific locations in the chromosome string.
- An allele is the possible values a gene can take (0/1 if binary, a-z if alpha, 0-9 if decimal).

Computational Model

- Crossover exchanges substrings between chromosomes.
- Mutation replaces a gene value with another from its allele.
- Inversion swaps the head with the tail of the chromosome at a locus.

Computational Model

Main GA algorithm

- Initial Population
- Evaluate
- Select fittest
- Recombine
- Generate initial population of candidate solutions
- Apply fitness function to population members
- Choose the fittest members to form new population
- Apply Genetic Operators and generate new population

Computational Model

The computational equivalent

A sample iteration

Questionnaire 1
Example: Search space & fitness landscapes

GAs versus other search methods

“Search” for what?

Data - Efficiently retrieve a piece of information, (Data mining) → Not AI

Paths to solutions - Sequence of actions/steps from an initial state to a given goal, (AI-tree/graph search)

Solutions - Find a good solution to a problem in a large space (search space) of candidate solutions
  - Aggressive methods (e.g. Simulated Annealing, Hill Climbing)
  - Non-aggressive methods (e.g. GAs)

Tree (Graph) Search

Solution: Sequence of steps/Path through graph

Solution "built" gradually (during graph traversal)

Exhaustive search (constraints-assisted by heuristics)

E.g. Depth-first, Breadth-first, Branch&Bound, ...

Search for solutions aggressively

Solution discovered gradually

Problem: Can be trapped in local maximum

Discovers a hilltop

E.g. Steepest Ascend

Search for solutions with GAs

Solution discovered probabilistically

Problem: Can not guarantee discovery of hilltop

Traces global maxima (wanders in whole search space)

Combined w/ aggressive algorithm can find global maximum

Why evolution?

Evolution is a massively parallel search method

Many computational problems require searching through a huge number of possibilities for solutions

Evolution use continually changing fitness criteria as creatures evolve

Many computational problems require adaptive solutions that perform well in changing environments

Evolution is remarkably simple, yet responsible for extraordinary variety and complexity

Many computational problems require complex solutions that are difficult to program by hand
Applications of GAs

- Numerical and Combinatorial Optimisation
  - Job-Shop Scheduling, Traveling salesman
- Automatic Programming
  - Genetic Programming
- Machine Learning
  - Classification, NNet training, Prediction
- Economic
  - Bidding strategies, stock trends
- Ecology
  - Host-parasite coevolution, resource flow, biological arm races
- Population Genetics
  - Viability of gene propagation
- Social systems
  - Evolution of social behavior in insect colonies

Application: Genetic Programming (GP)

- Automatic programming implies the existence of computer programs that write ... computer programs
- Early work on Evolutionary Computation (Evolutionary programming) aimed at automatic programming
- GP [Koza '92, '94] used GAs for automatic programming
  - Evolve computer programs rather than write them

Genetic Programming: Overview

- Problem: We want to develop a system that builds programs that solve math equations
- Consider an instruction set for a zero-address VM (only stack, no registers)
- Solution encoding using byte strings for instruction strings:
  - Solution: OVER,ADD,MUL,ADD
  - Chromosome: 5 4 3 4

Genetic Programming: Overview

- Fitness Evaluation: for random args calculate
  \[ E = \text{Expected value} - \text{Executed instr stream} \]
- Recombination operators:
  - Crossover: break parent instruction streams at random point and exchange tails
  - Mutation: choose random position, replace instruction (gene) with another from the instruction set

GP: The (pseudo-) code

```
Main ( )
    noPopulation ( )
    max_fitness := 0
    foreach member chromosome
        fitness := EvaluateFitness (chromosome)
        if fitness > max_fitness
            max_fitness := fitness
            fittest_solution := chromosome
    while generation < MAX_GENERATIONS
        offspring := SelectAndRecombine (parents)
        fitness := EvaluateFitness (offspring)
        if fitness > max_fitness
            max_fitness := fitness
            fittest_solution := offspring
    print fittest_solution
```
GP: The (pseudo-) code

```plaintext
InitPopulation ()
   while num_of_programs < MAX_PROG
      command_list := RandomSelectCommands ()
      new_member := Concatenate (commands_list)
      SelectAndRecombine ()
   while num_of_programs < MAX_PROG/2
      parent_1 := RouletteWheelSelection ()
      parent_2 := RouletteWheelSelection ()
      randomly choose x-over point
      child_1 := parent_1 [head] ^ parent_2 [tail]
      child_2 := parent_2 [head] ^ parent_1 [tail]
      foreach child
         mutation_pt := RandomlyChooseMutationPoint (child)
         new_instr := RandomlyChooseNewInstruction ()
         ReplaceInstruction (mutation_pt, new_instr)
      Evaluate_fitness ()
      forall chromosomes in population
         repeat COUNT times
            args := GenerateRandomArgs ()
            expected_result := SolveEquation (args)
            InterpetFSM (chromosome, args)
            if STACK_OVERFLOWN
               fitness += TIER1
            else if stack[0] != expected_result
               fitness += TIER2
            else
               fitness += TIER3
            avg_fitness := fitness of all chromosomes / MAX_PROG
   Endfor
```

GP: The (pseudo-) code

```plaintext
Evaluate_fitness ()
   forall chromosomes in population
      repeat COUNT times
         args := GenerateRandomArgs ()
         expected_result := SolveEquation (args)
         InterpetFSM (chromosome, args)
         if STACK_OVERFLOWN
            fitness += TIER1
         else if stack[0] != expected_result
            fitness += TIER2
         else
            fitness += TIER3
         avg_fitness := fitness of all chromosomes / MAX_PROG
   Endfor
```

GP: The (pseudo-) code

```plaintext
InterpetFSM ()
   push args in stack and increase stack ptr
   while program counter < program_length
      result := ExecuteCommand (args)
      push result in stack
   Endfor
```

Genetic Programming: Results

- Results of a few runs
  - x^8 :
    DUP, MUL, DUP, MUL, DUP, MUL, MUL -> ((x*x) * (x*x) * (x*x) * (x*x) * (x*x) * (x*x) * (x*x))
  - 2x + 2y + z :
    ADD, DUP, ADD, SWAP, ADD -> ((x+y) + (x+y)) + z
  - xy + y^2 + z :
    OVER, ADD, MUL, ADD -> ((x*y) + y^2) + z
  - x^2 + y^2 + z :
    DUP, DUP, MUL, MUL, SWAP, DUP, MUL, SWAP, ADD, SWAP, SWAP, ADD -> ((x*x) + (y*y) + z)
```

Genetic Programming: Performance

- Chromosome representation must capture the dependencies of genes
- Initial population must be diverse
- Selection must make sure the fittest chromosomes are propagated to the next population and at the same time maintain diversity
- Recombination should not destroy good genes
- Mutation must guarantee seeding of new genetic material
Some References – GA/GP

- **GA Seminal paper**

- **GA/GP Books**
  - Chapter D. H. Ballard, 1999: *An Introduction to Natural computation*, MIT Press

- **Artificial Intelligence course textbooks**

- **GAs on-line tutorial**
    [http://samizdat.mines.edu/ga_tutorial/ga_tutorial.ps](http://samizdat.mines.edu/ga_tutorial/ga_tutorial.ps)

- **GP on-line resources**

Questions ...

Some References – GA/GP

- **Something different !**

TSP with GAs