



Genetic Algorithms

CSc355

Alan Dix

dixa@comp.lancs.ac.uk

Manolis Sifalakis

mjs@comp.lancs.ac.uk

Lecture Overview

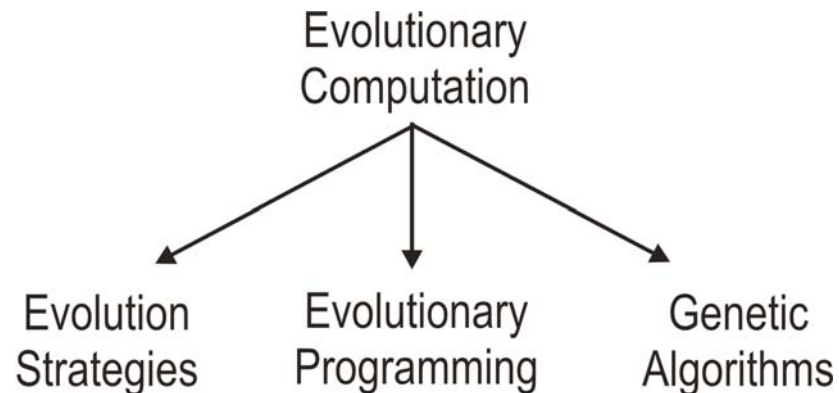
- Evolutionary Computation (EC) and Genetic Algorithms (GA)
 - A Brief History
 - Terminology from Biology
 - Computational Model
 - Search space and fitness landscapes (an example)
 - GAs versus other search methods
 - Why evolution
 - Applications of GAs
 - Sample Application: Genetic Programming (GP)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



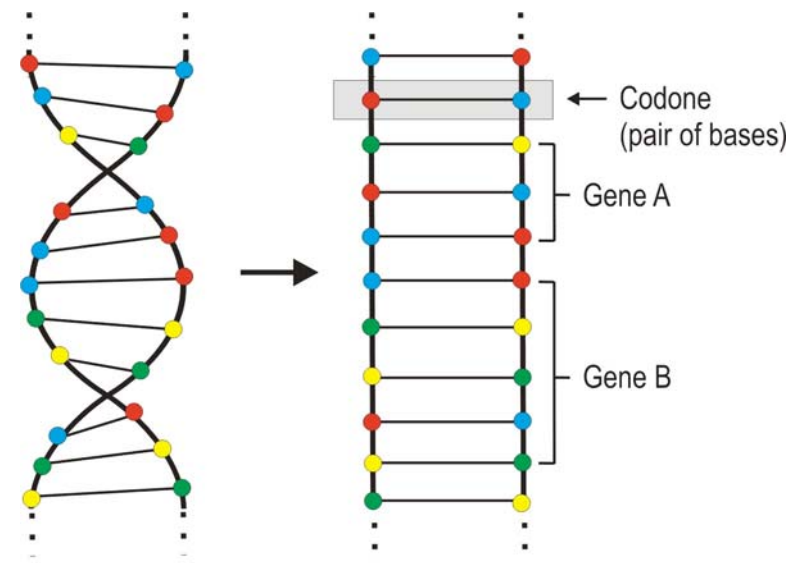
A Brief History



- **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 [Holand 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 (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

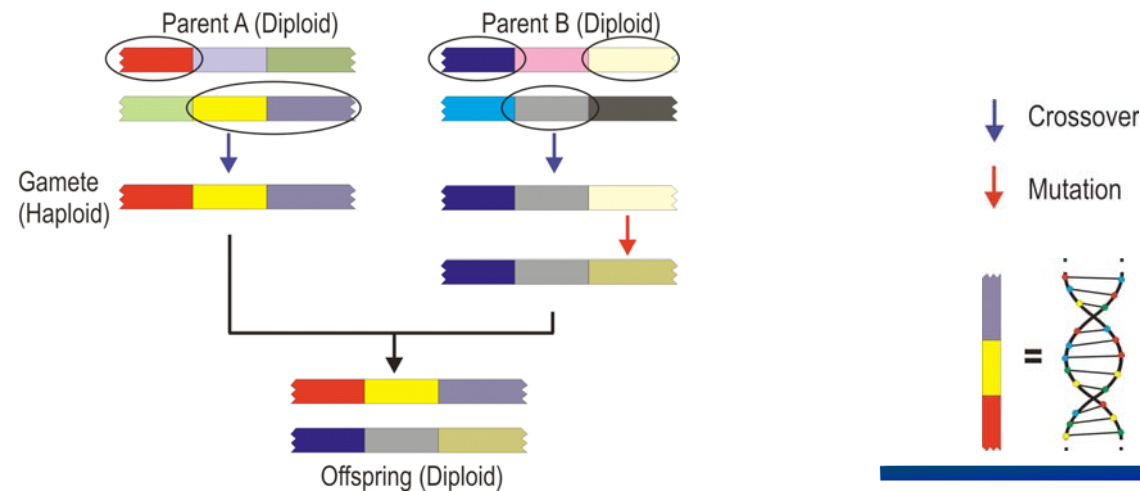


Biological Systems: A rough guide

- **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 and 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

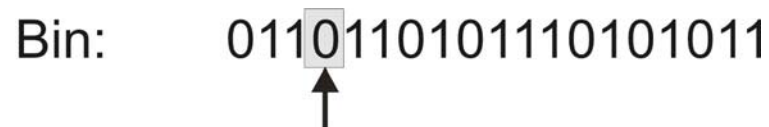
- A **chromosome** is a string representation of a **candidate solution** to a problem

Bin: 0110110101110101011

Alpha: AABCAABCCDGGABCD

Hex: 937ff4539acc27d4bb92

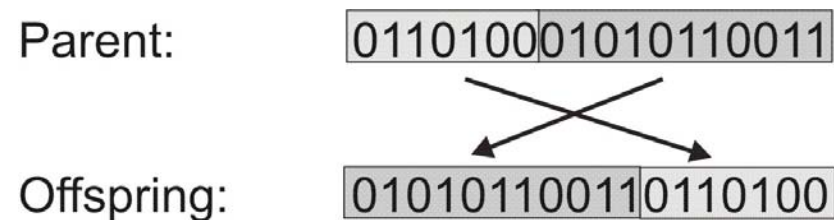
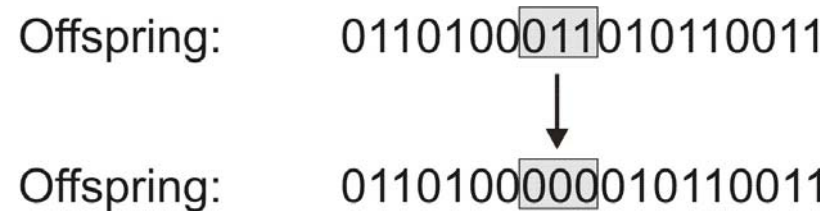
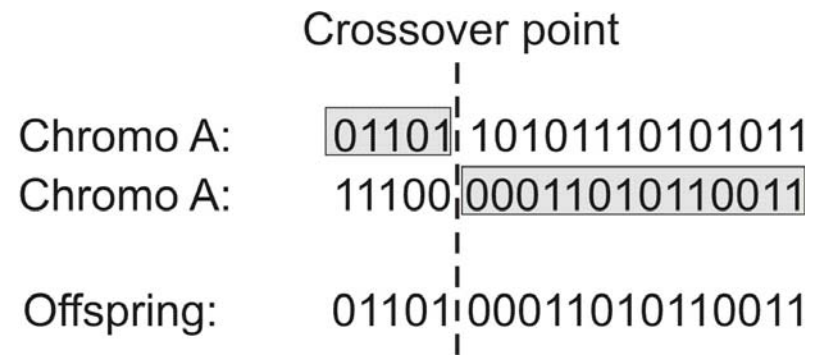
- The **genes** are **single digits or subsets of digits** at specific locations in the chromosome string

Bin: 0110110101110101011


- 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

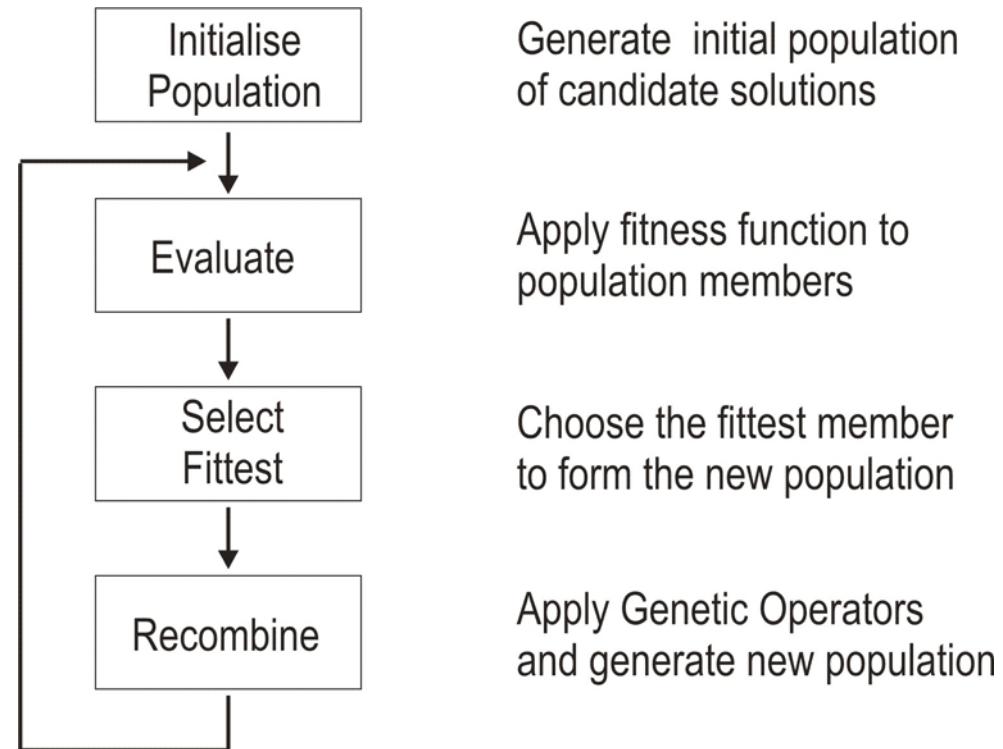


Questionnaire 1



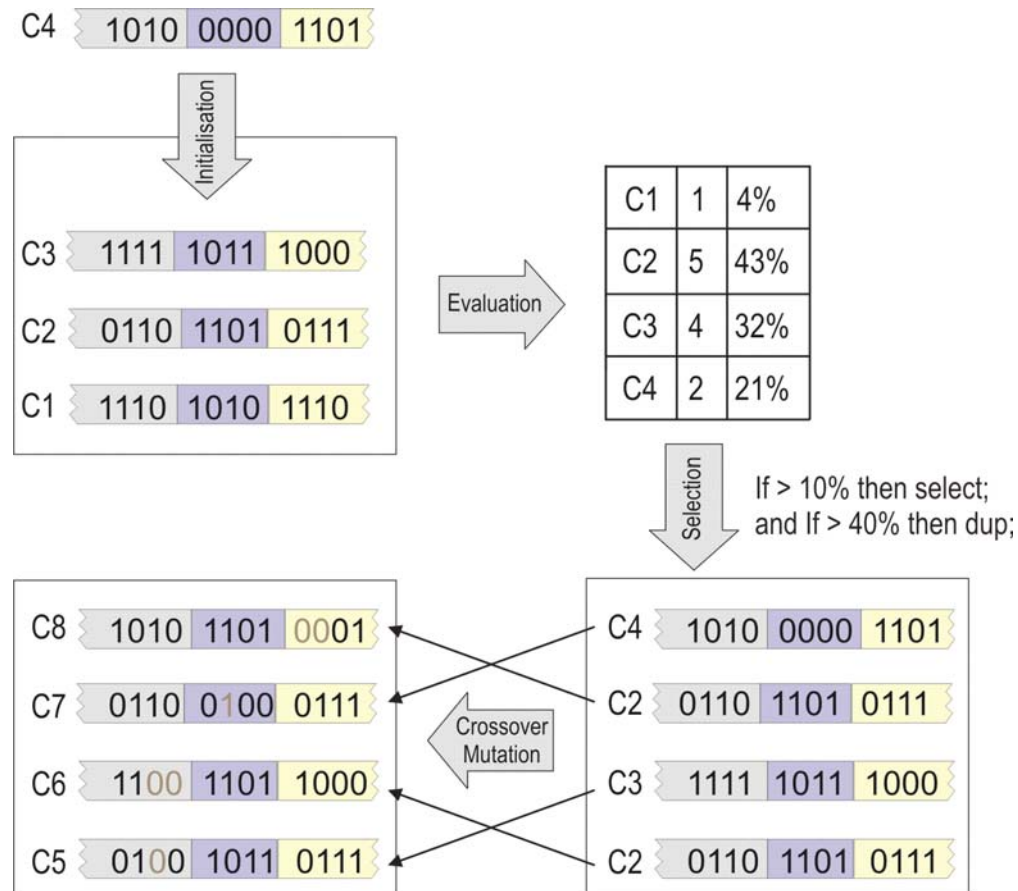
Computational Model

Main GA algorithm



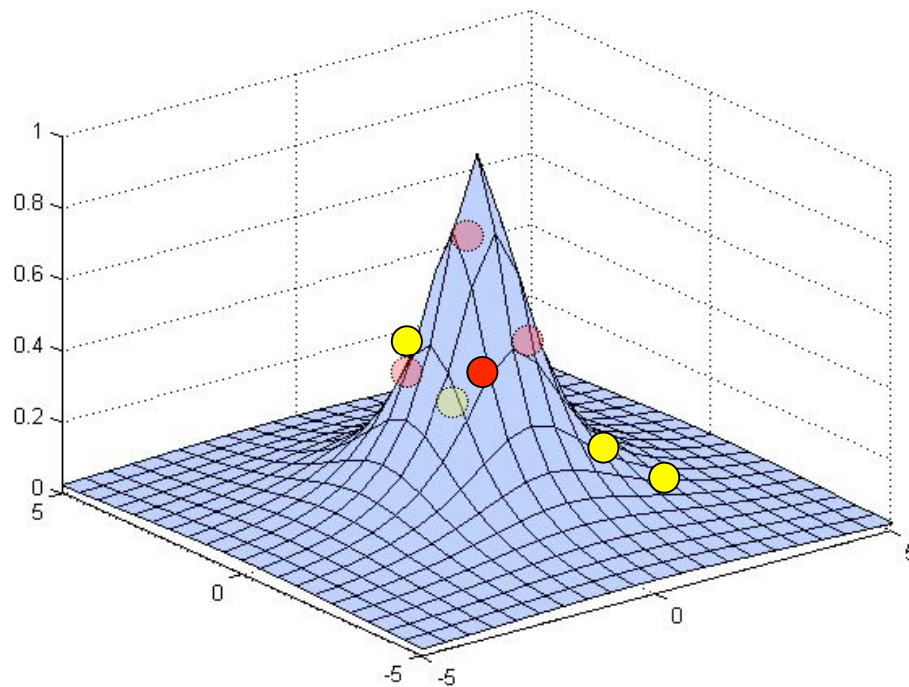
The computational equivalent

A sample iteration



Example: Search space & fitness landscapes

$$F(x,y) = \frac{1}{1 + x^2 + y^2}$$



	X	Y
C1	-1	2
C2	-2	3
C3	1.5	0
C4	0.5	-1

Chromosome Encoding: Cartesian Coords

X coord Y coord

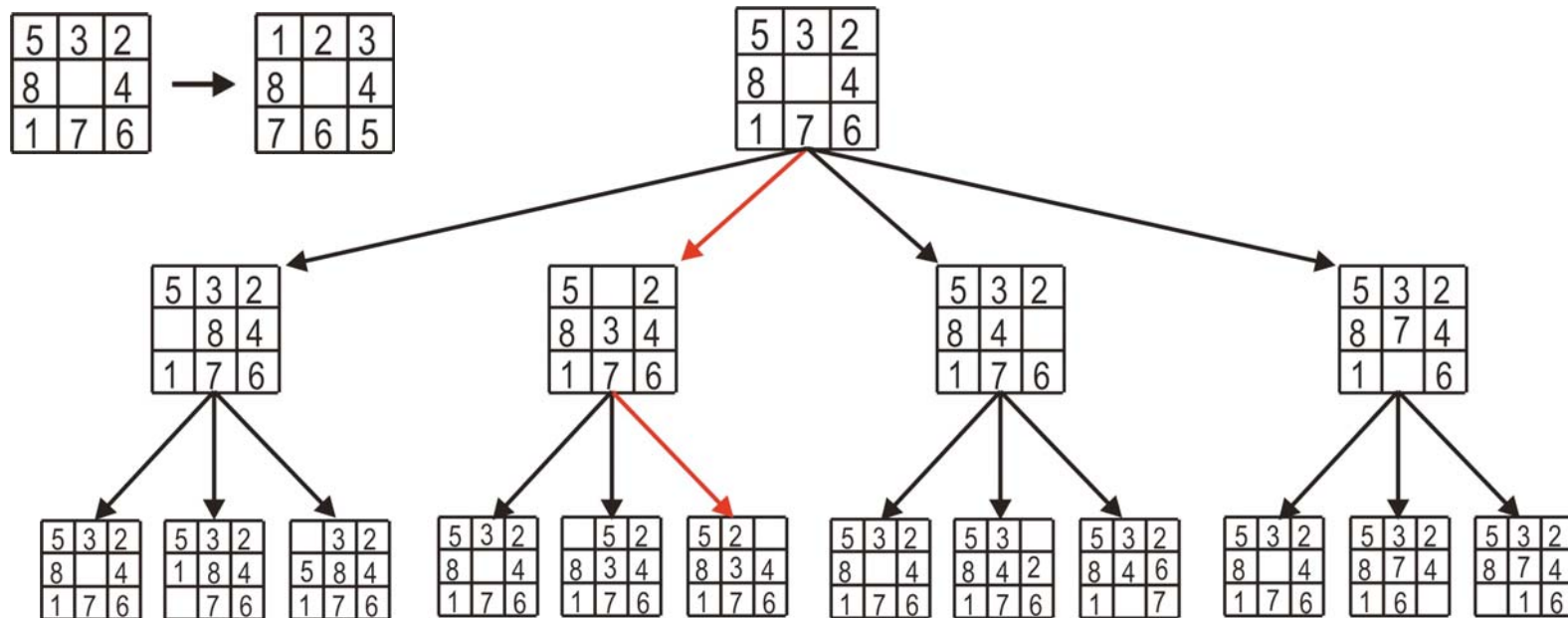
GAs versus other search methods

“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)
-

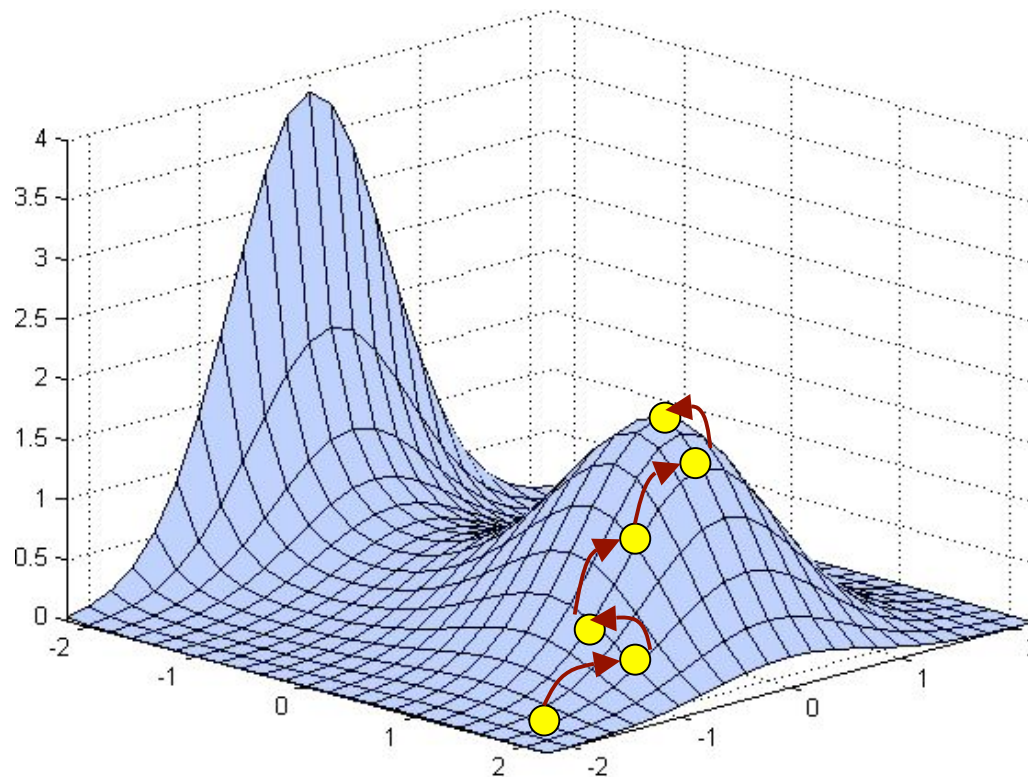
Tree (Graph) Search

- 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, ...

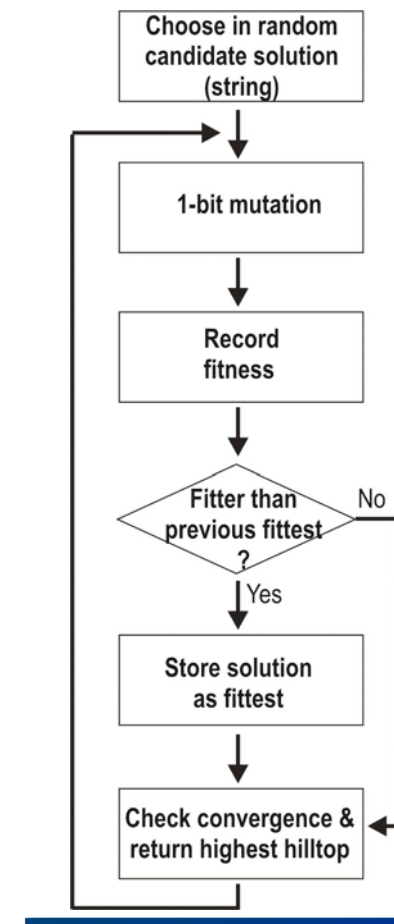


Search for solutions aggressively

- Solution discovered gradually
- Problem: Can be **trapped in local maximum**
- Discovers a hilltop

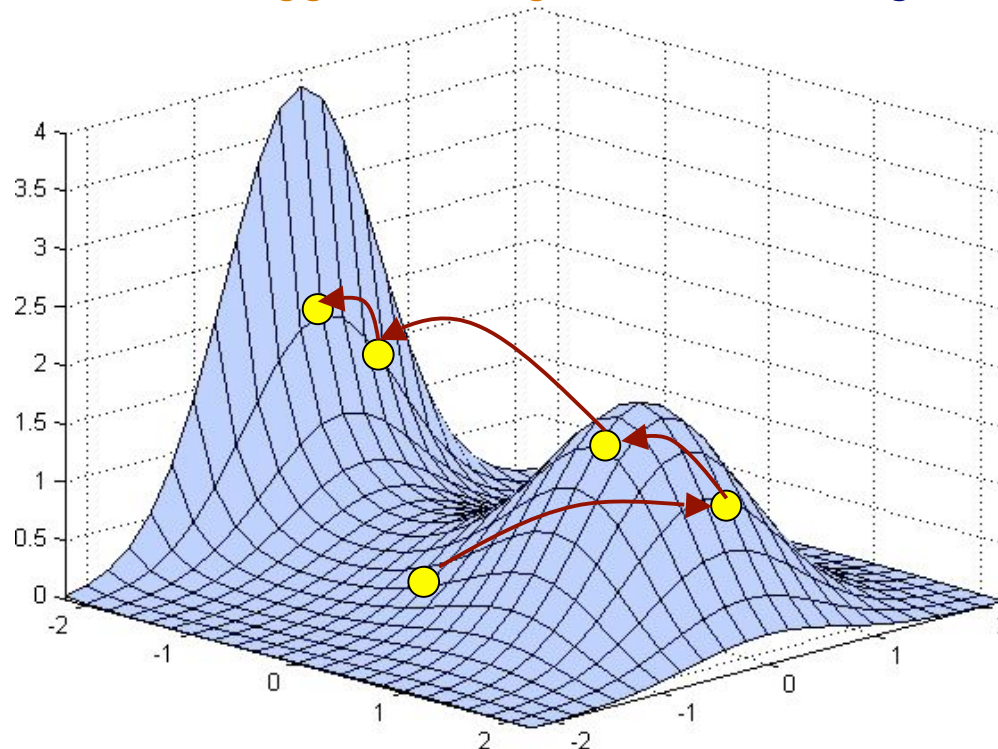


E.g. Steepest Ascend

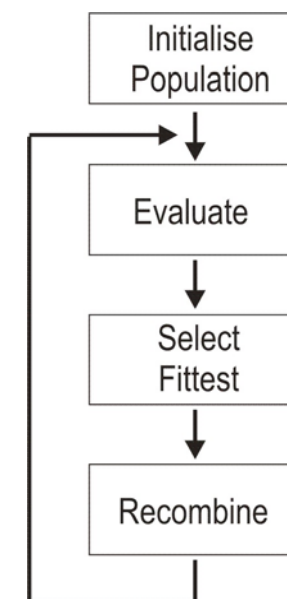


Search for solutions with GAs

- 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



Genetic Algo



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
-

Questionnaire 2



Applications of GAs

- Numerical and Combinatorial Optimisation
 - Job-Shop Scheduling, Traveling salesman
 - Automatic Programming
 - Genetic Programming
 - Machine Learning
 - Classification, NNet training, Prediction
 - Economic
 - Biding 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

Code	Operator	Description
1	DUP	$X \Rightarrow AA$
2	SWAP	$XY \Rightarrow YX$
3	MUL	$XY \Rightarrow (X*Y)$
4	ADD	$XY \Rightarrow (X+Y)$
5	OVER	$XY \Rightarrow XYX$
6	NOP	Null

Genetic Programming: Overview

- **Fitness Evaluation:** for random args calculate

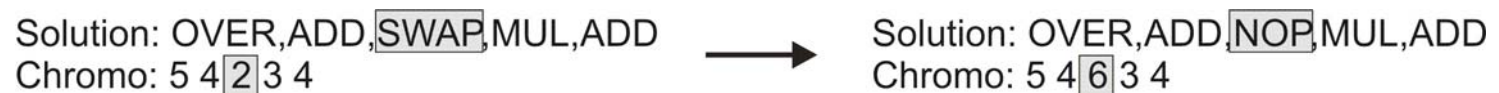
$E = \text{Expected value} - \text{Executed instr stream}$

- **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 ( )  
  InitPopulation ( )  
  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

InitPopulation ()

```
while num_of_programs < MAX_PROG  
  command_list := RandomSelectCommands ( )  
  new_member := Concatenate (command_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)
```

GP: The (pseudo-) code

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 // stack[0] == expected_result

fitness += TIER3

avg_fitness := fitness of all chromosomes / MAX_PROG

GP: The (pseudo-) code

InterpretFSM ()

push args in stack and increase stack ptr
while program counter < program_length
 pop args from stack
 result := *ExecuteCommand* (args)
 push result in stack

Genetic Programming: Results

- Results of a few runs

- x^8 :

DUP, MUL, DUP, MUL, DUP, MUL $\rightarrow ((x*x)*(x*x))*((x*x)*(x*x))$

- $2x + 2y + z$:

ADD, DUP, ADD, SWAP, ADD $\rightarrow ((x+y)+(x+y))+z$

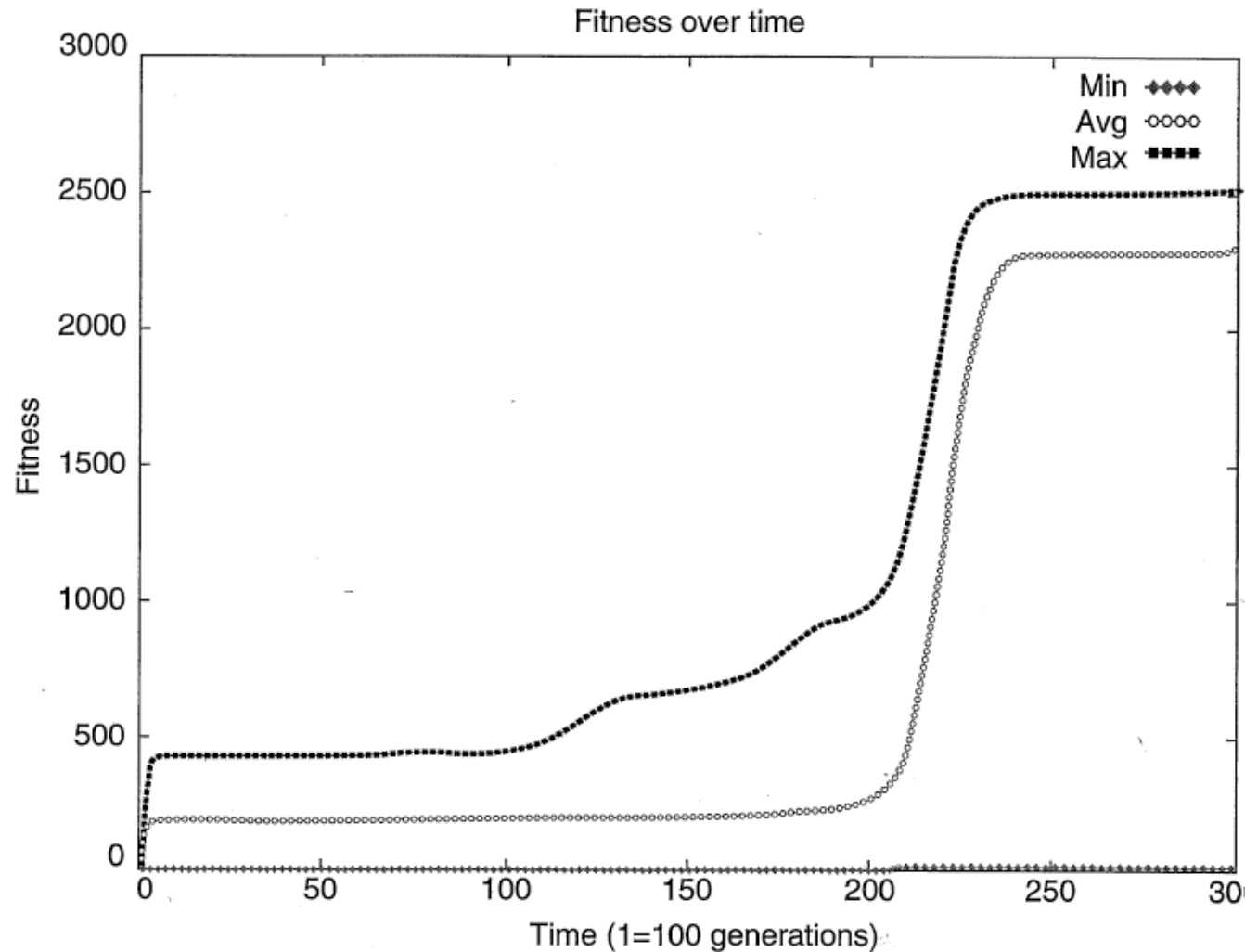
- $xy + y^2 + z$:

OVER, ADD, MUL, ADD $\rightarrow ((x+y)*y)+z$

- $x^3 + y^2 + z$:

DUP, DUP, MUL, MUL, SWAP, DUP, MUL, SWAP, ADD, SWAP, SWAP,
ADD $\rightarrow ((x*x*x)+(y*y))+z$

Genetic Programming: Performance



Performance of GAs

- 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
 - Holland J.H., *Adaptation in natural and artificial system*, Ann Arbor, The University of Michigan Press, 1975

- GA/GP Books
 - M. Mitchell. 1998: *An Introduction to Genetic Algorithms*. MIT Press
 - J. Koza. 1992: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press
 - *Chapter*: M. Tim Jones. 2003: *AI Application Programming*. Charles River Media, Inc
 - *Chapter*: D. H. Ballard. 1999: *An Introduction to Natural computation*. MIT Press

- Artificial Intelligence course textbooks

- GAs on-line tutorial
 - Darell Whitley, *An Introduction to Genetic Algorithms*.
http://samizdat.mines.edu/ga_tutorial/ga_tutorial.ps

- GP on-line resources
 - <http://www.genetic-programming.com/>

Some References – GA/GP

- Something different !
 - R. Dawkins, 1976: *The Selfish Gene*. Oxford University Press.
 - S. Blackmore, 1999: *The Meme Machine*. Oxford University Press.

Questions ...



TSP with GAs

