Search and Genetic Algorithms

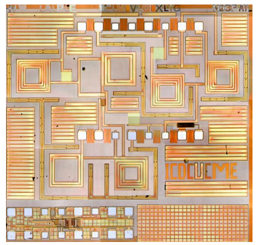
Stephen Marsland

Optimisation

- Most of this course has been about optimisation
  - Gradient descent
- What if there is no gradient?
  - Discrete problems
  - Check all cases?

Discrete Problems

- Chip Design
  - Position circuits on a chip so that none of the lines cross
- Timetabling
  - Given courses and students, find a timetable with the minimum number of clashes



No Free Lunch

- There is no perfect search algorithm
- You always need to think about the problem
- Might have to design the encoding of the problem carefully

- In practice, gradient descent is pretty good for continuous problems

The Travelling Salesman

Find a tour that starts and ends at the same city, visits every city precisely once, and has the minimum possible distance

The Travelling Salesman

- An NP-complete problem
- There are N! different possible solutions (where N is the number of cities)
  - Virtually impossible to solve for N>10
- Actually useful - logistics, chip design

Exhaustive Search

- Try out every solution
- Trivial to implement
- O(N!) worse than O(exp(N))

Greedy Search

- Choose a start city
- Repeat
  - Pick the closest remaining city
- Until get back to start city

- No backtracking

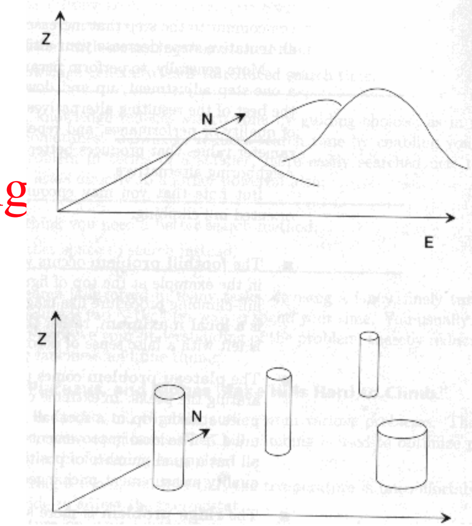
Greedy Search

- Computationally cheap: O(N)
- Will find a solution
- No guarantee of any kind of optimality
- Cannot predict how good solution is

Hill Climbing

- Pick a starting cycle
- Repeat
  - Swap a random pair of cities
  - If tour length decreases keep the new order
- Until you get bored or stops changing

Hill Climbing



Hill Climbing

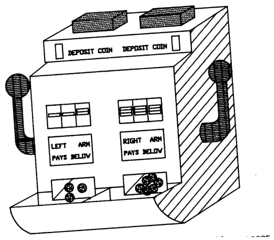
- Foothills
  - Local maxima
- Plateaus
  - Algorithm fails to get anywhere at all
- Ridges
  - Look like peaks - most directions go down

Exploitation and Exploration

- Try out new solutions
  - Exploration
  - Exhaustive search
- Try to improve your current best solution
  - Exploitation
  - Hill climbing
- Ideally, some of both

Exploitation and Exploration

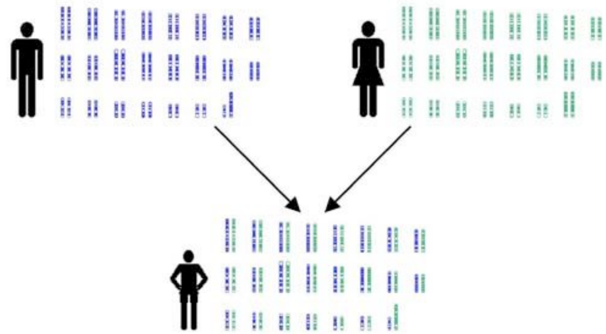
- Room of 1-armed bandits
- Maximise your payout
- Do you
  - Use the best machine you've found so far
  - Try out some more hoping to find a better one



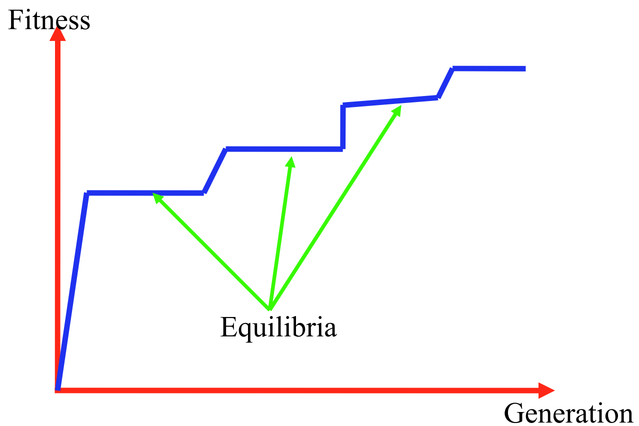
Evolution

- Survival of the fittest
  - Live longer (more chance to reproduce)
  - More attractive (more chance to reproduce)
  - Higher number of offspring (more survive)
- 50% chance of any gene being in offspring

Evolution



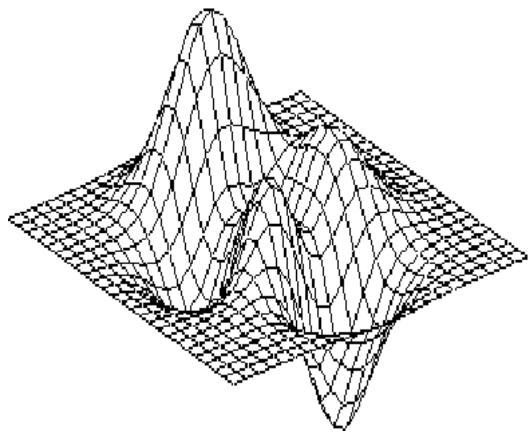
Punctuated Equilibrium



Fitness Landscapes

- Useful imagination aid
- Animals fitness is partly due to competition with other animals and environment
- Changes over time
- Evolution favours animals that evolve to peaks of the fitness landscape

Fitness Landscapes



The Genetic Algorithm

The Genetic Algorithm

- How can we abstract the useful bits of evolution?
- How can we fit them into an algorithm?
- Does it work?

Representation: Strings and Things

Chromosome	String
Gene	Elements
Allele	Alphabet

Representation: Strings and Things

- Work out a way of encoding problem
- Choose an alphabet
  - Possible values of each element of string
  - Often binary
- Not always easy
- Split up the problem into discrete parts

Representation: Strings and Things

- Example: bill paying
- List of 100 bills to pay
- Use string of 100 elements
- Each element is whether to pay one bill
- 10110 means pay bills 1, 3, and 4

Fitness Functions

- Decide how good the string is
- You pass in a string and get back a number
- The higher the number, the better the solution
- Generally a positive number
- Problem-specific part

Selection

- Choosing parents is crucial
- Want the best (fittest) strings to reproduce
  - Exploitation
- What about non-fit strings?
  - Exploration
- Generate a ‘mating pool’

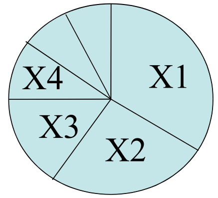
Selection Methods

- If use uniform distribution, all strings equally likely
- Probably not good
- Fitness proportional selection
  - Pick proportional to fitness

$$p^{\alpha} = \frac{F^{\alpha}}{\sum_{\alpha'} F^{\alpha'}}$$

Selection Methods

- Like a roulette wheel
- Probability of picking a string is proportional to its area on wheel
- Fitter strings have a larger number of copies - larger area



Selection Methods

- Truncation selection
  - Pick the top 50% of strings
  - Choose from them at random
- Elitism
  - Keep a copy of the best strings all the time
- Tournaments
  - Put the fittest 2 out of the parents and offspring into the new population

Selection Methods

- Niching
  - Sometime exploration stops - premature convergence
  - Evolve several subpopulations and occasionally swap a few elements
  - Also called island populations

Genetic Operators

- How do we combine the two parents?

1 0 0 1 1 0 0 0 1 0 1
0 1 1 1 1 0 1 0 1 1 0

Genetic Operators

- Single Point Crossover

1 0 0 1 1 0 0 0 1 0 1
0 1 1 1 1 0 1 0 1 1 0

---

1 0 0 1 1 0 1 0 1 1 0

Genetic Operators

- Multi-Point Crossover

1 0 0 1 1 0 0 0 1 0 1
0 1 1 1 1 0 1 0 1 1 0

---

1 0 0 1 1 0 1 0 1 0 1

Genetic Operators

- Uniform Crossover

Random Samples
0 0 1 1 0 1 1 0 1 1 0

String 0 1 0 0 1 1 0 0 0 1 0 1
String 1 0 1 1 1 1 0 1 0 1 1 0

---

1 0 1 1 1 0 1 0 1 1 1

Genetic Operators

- Mutation

1 0 1 1 $\boxed{1}$ 0 1 0 1 1 1

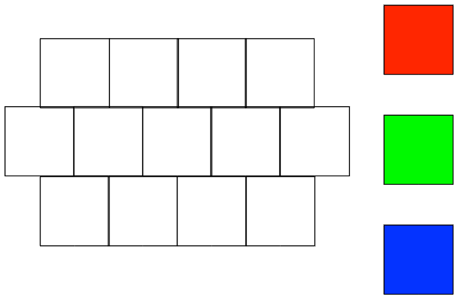
1 0 1 1 0 0 1 0 1 1 1

Putting It All Together

- Generate a random population of strings
- Repeat
  - Compute their fitnesses
  - Select the mating pool
  - Crossover parents to produce offspring
  - Mutate the results
  - Put them in the new population
- Until stopping criteria met

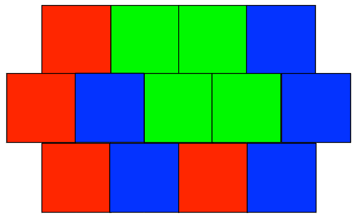
Genetic Algorithms in Practise: Colouring A Map

The Problem



- Colour the map with any of three colours
- No two bordering countries can have the same colour

Making a GA for 3-Colouring

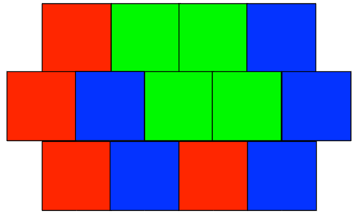


- Represent possible solutions as strings

RGGBRBGGBRBRB

Making a GA for 3-Colouring

- Devise a fitness function
  - Fitness increases for better solutions
  - F = number of boundaries that are OK



16 out of 26 in this example

Making a GA for 3-Colouring

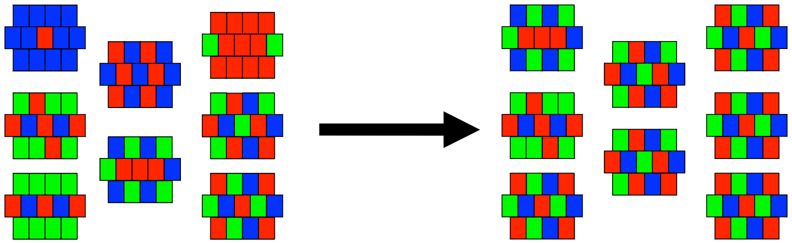
- Make a population of solutions

[R G B B …]   [G G G R …]   [R G R B …]

[G B R R …]   [B G R B …]   [G B R G …]

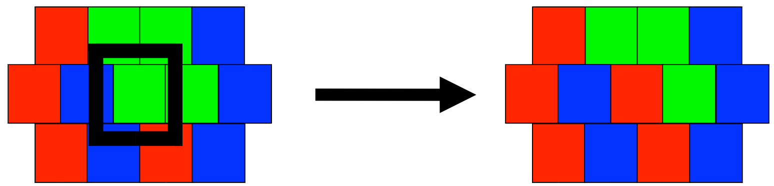
- Apply genetic operators to them to make a new population
- Iterate

Making a GA for 3-Colouring



- Generate a new population from the old one
- Bias the selection to pick the fitter strings more often

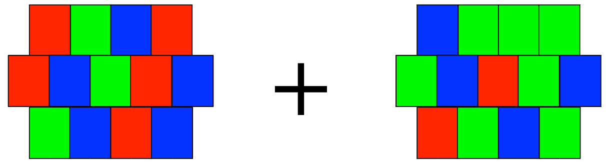
Making a GA for 3-Colouring



Mutation

- Change randomly selected bit to another colour with small probability

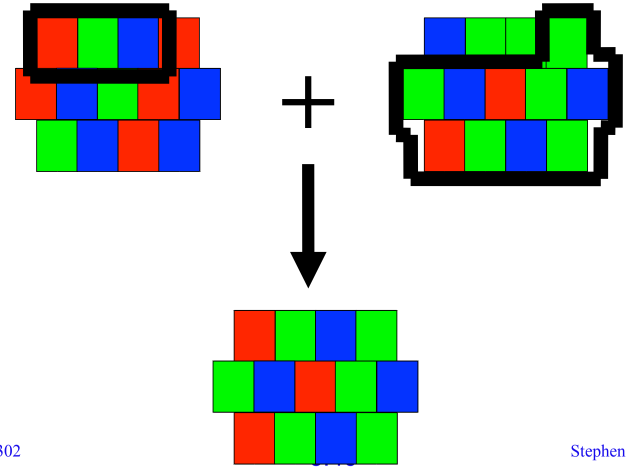
Making a GA for 3-Colouring



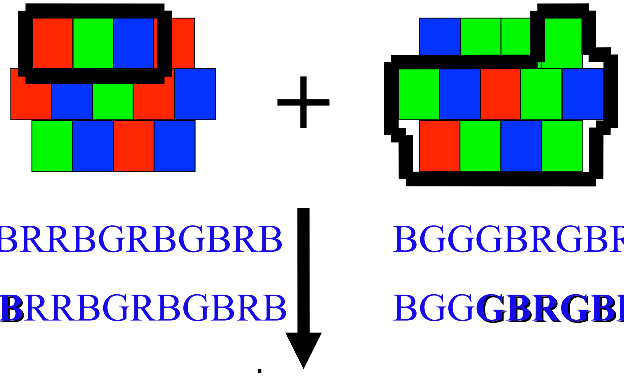
Crossover

- Combine pairs of strings to form new strings

Making a GA for 3-Colouring



Making a GA for 3-Colouring



RGBRRBGRBGBRB

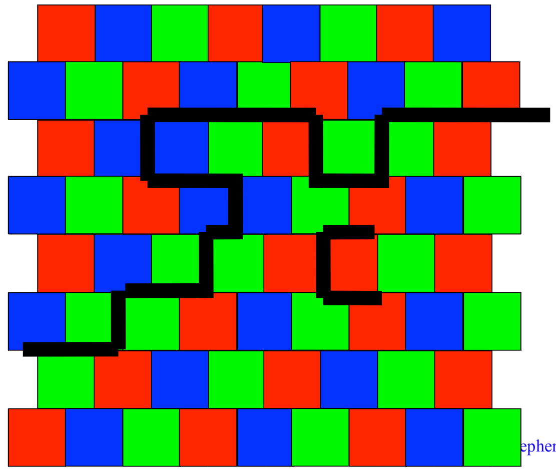
RGBRRBGRBGBRB

BGGGBRGBRGBG

BGG**GBRGBRGBG**

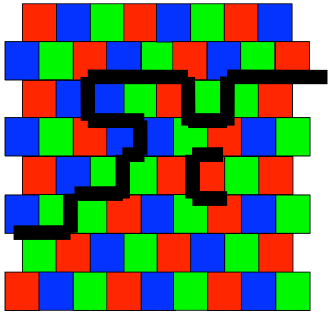
RGB¦GBRGBRGBG

Making a GA for 3-Colouring

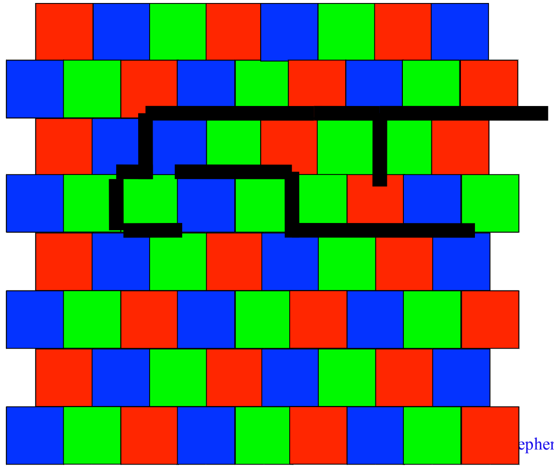


How Important Is Crossover?

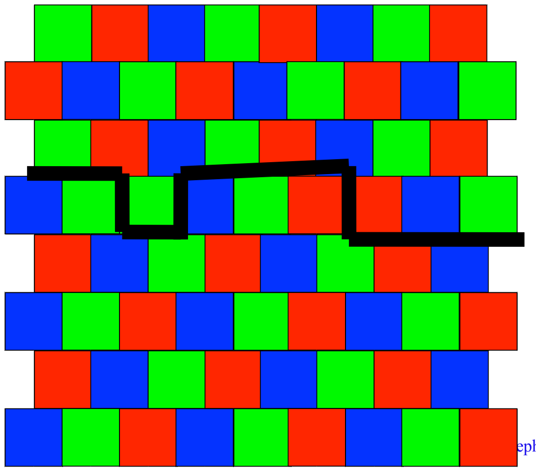
Making a GA for 3-Colouring



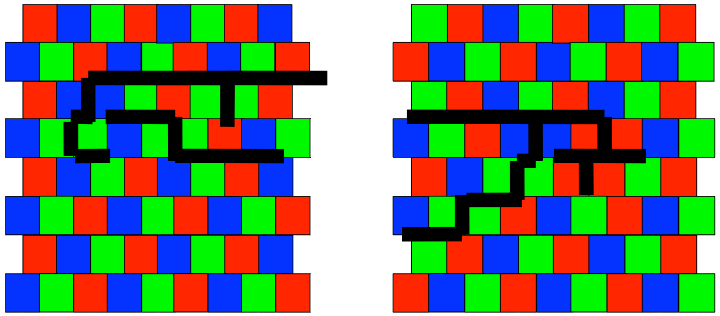
Making a GA for 3-Colouring



Making a GA for 3-Colouring



Making a GA for 3-Colouring



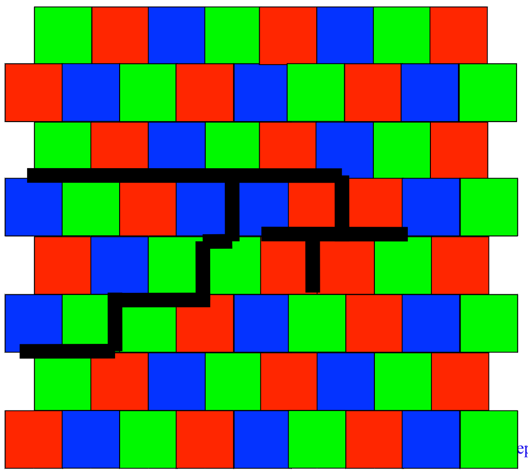
Premature Convergence

- Fitter members of population are favoured
- Solutions at local maxima are favoured
  - Exploitation, not exploration
  - Selection for the local maximum
  - Diversity in population reduces
- Already seen niching
- Can also average fitness across number of identical strings (fitness sharing)

Neural Networks and Genetic Algorithms

- Multi-Layer Perceptron
  - Computed error at each neuron
  - Computed gradient
- Could use GA instead of gradient descent
  - Throw away gradient information
  - Replace all errors by 1 fitness

Making a GA for 3-Colouring



Guaranteed Convergence?

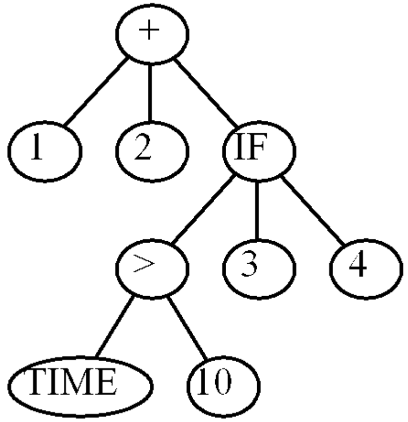
- Not much successful analysis
  - Not guaranteed to converge at all
  - Not guaranteed to reach global maximum
  - Can be very slow
- Trade-off between the genetic operators

Neural Networks and Genetic Algorithms

- More sensible: use GA to select the network structure
  - Mutation only - crossover not sensible
  - Several mutations:
    - ✓Add a node
    - ✓Delete a node
    - ✓Add a weight connection
    - ✓Delete a weight connection
  - Use normal training

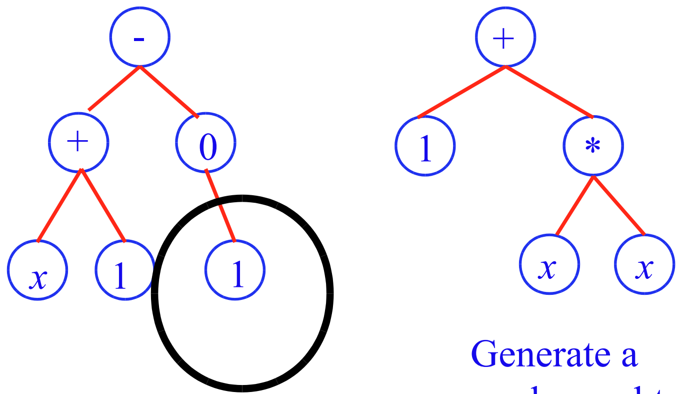
Genetic Programming

- Represent computer programs as trees
- Evolve the trees
- Fitness is how well/quickly the program works
- Used for many tasks
  - Skin melanoma detection
  - Chip design



`(+ 1 2 (IF (> TIME 10) 3 4))`

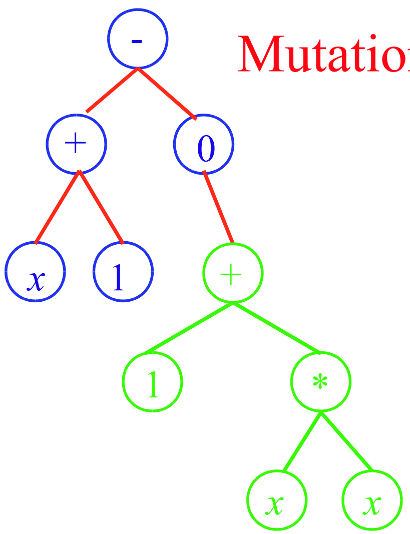
Mutation



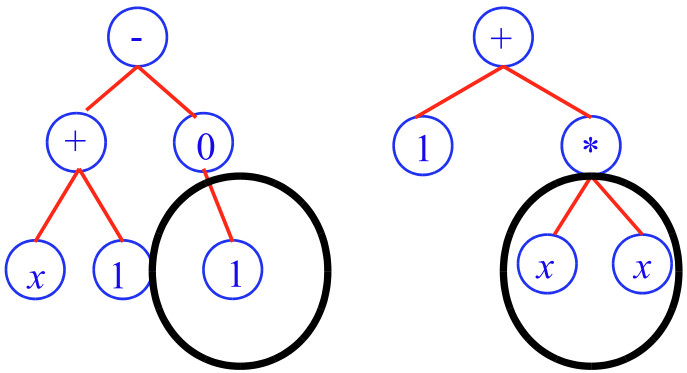
Choose a subtree to replace

Generate a random subtree

Mutation

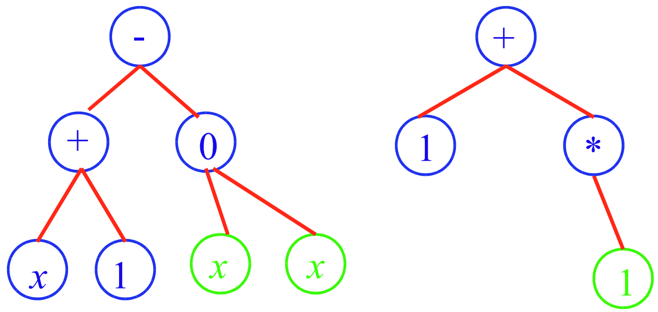


Crossover



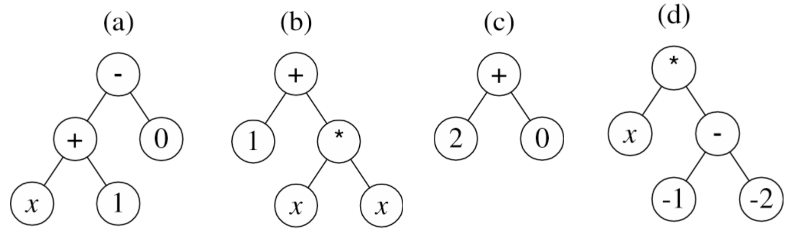
Choose a pair of subtrees to swap

Crossover



Genetic Programming

4 randomly chosen initial trees

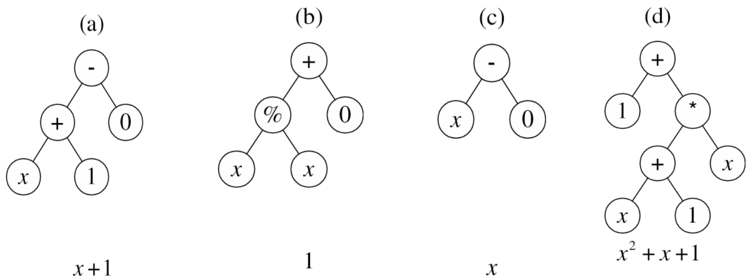


$x+1$ x^2+1 2 x

Genetic Programming

- Search space unbelievably large
- Depends strongly on initial population
- Programmer develops a set of possibly useful subtrees

Genetic Programming



Copy of (a)

Mutation of (c) from the (2)

Crossover of (a) and (b)

Second crossover of (a) and (b)