Genetic Algorithms

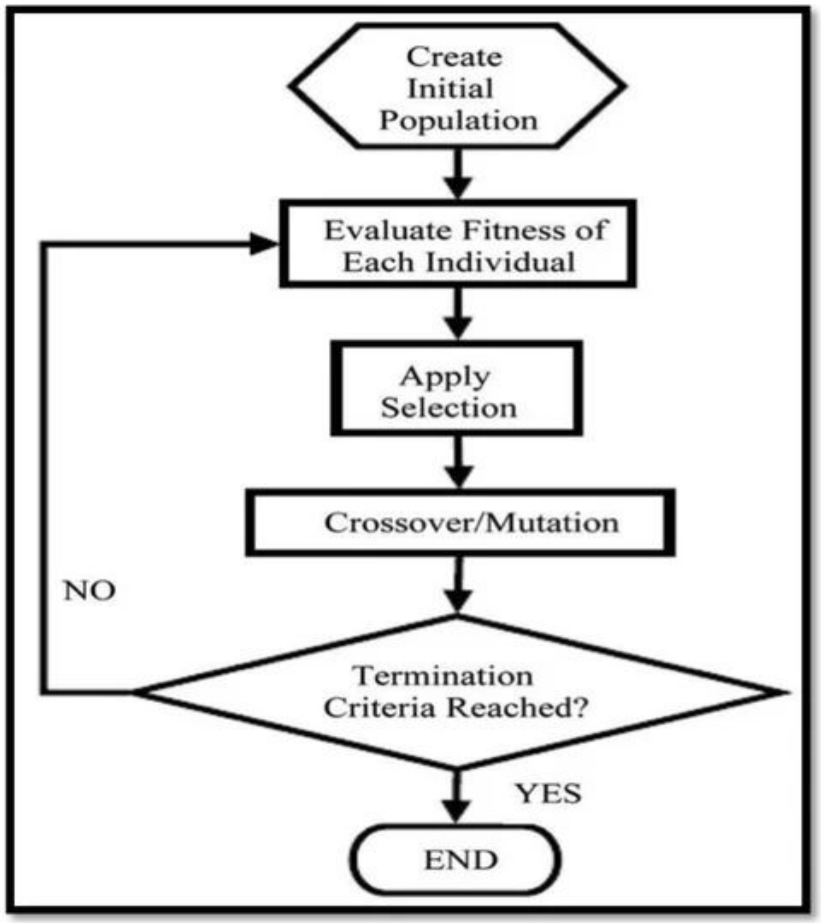
Principles of natural selection applied to computation:

- Variation
- Selection
- Inheritance

Evolution in a computer:

- Individuals (genotypes) stored in computer's memory as bit strings
- Evaluation of individuals (artificial selection) with a fitness function
- Differential reproduction through copying and deletion
- Variation introduced by analogy with mutation and crossover

Genetic Algorithms

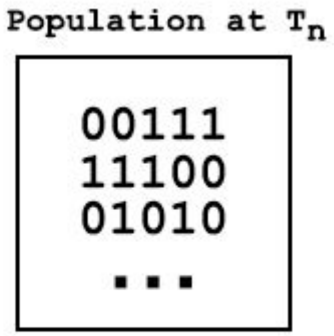

Create Initial Population
Evaluate Fitness of Each Individual
Apply Selection
Crossover/Mutation
NO
Termination Criteria Reached?
YES
END

Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach

by Ahmad Hassanat [1,2,*,†,‡], Khalid Almohammadi [1], Esra'a Alkafaween [2,‡], Eman Abunawas [2], Awni Hammouri [2] and V. B. Surya Prasath [3,4,5,6]

How it Works

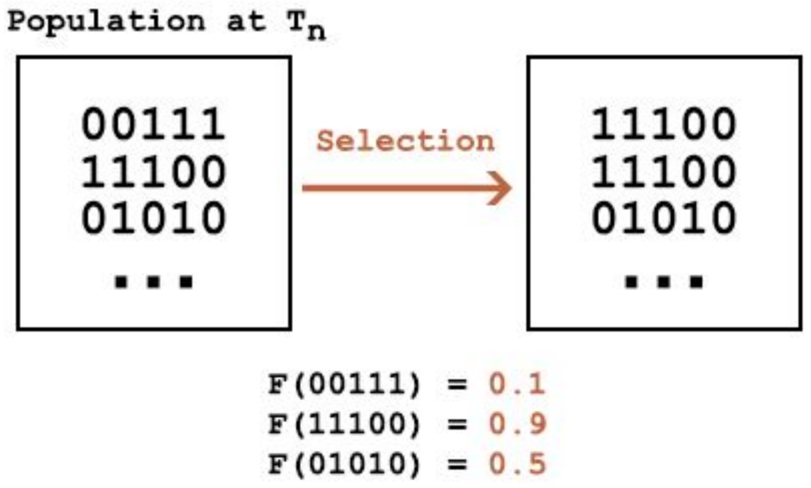
1. A population of individuals is created in a computer.



Slide credit: Justin Carmichael and Sonny Trujillo Jr.

How it Works

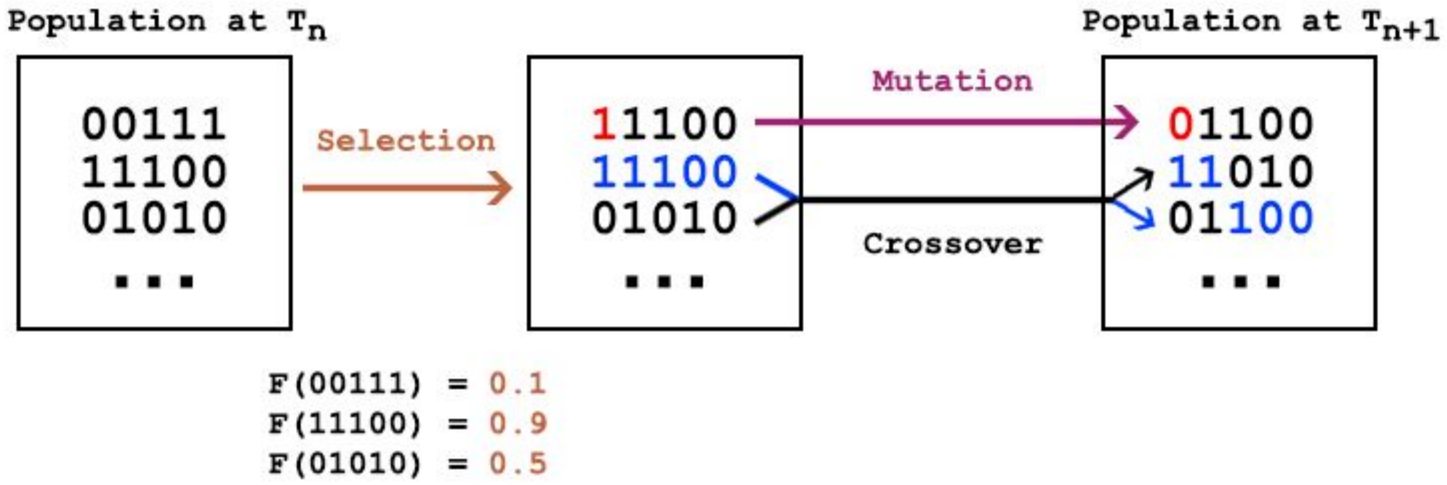
1. A population of individuals is created in a computer.
2. Individuals are selected based on fitness



Slide credit: Justin Carmichael and Sonny Trujillo Jr.

How it Works

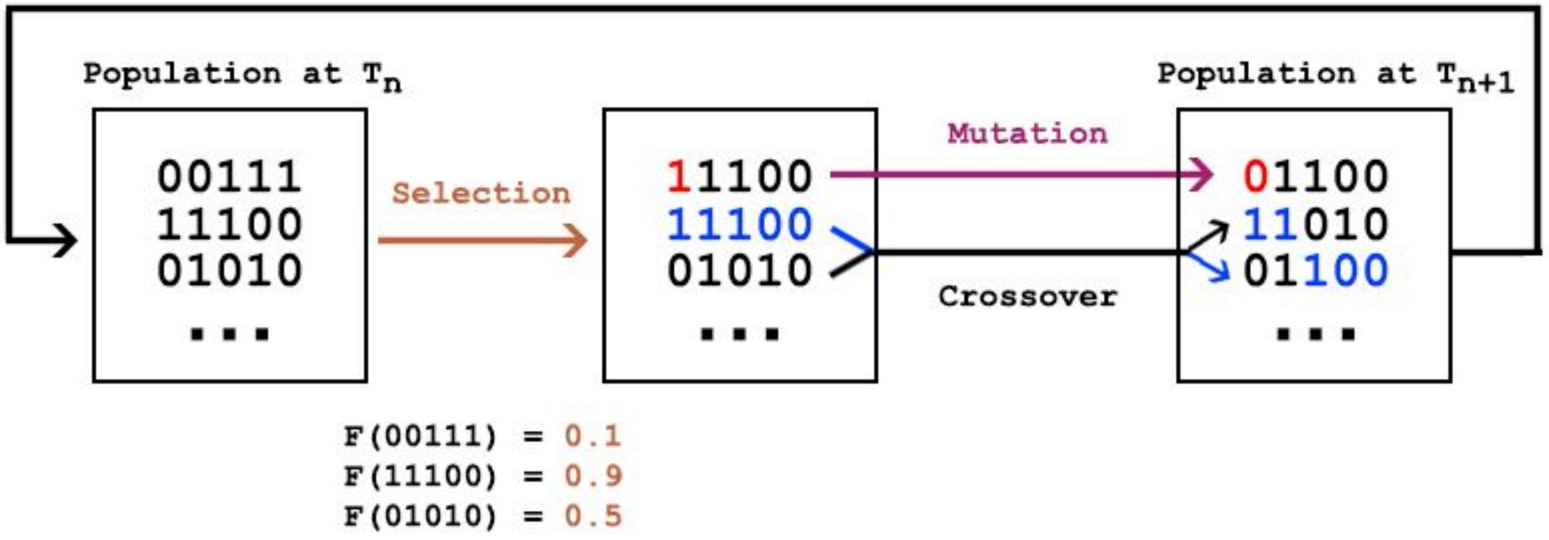
1. A population of individuals is created in a computer.
2. Individuals are selected based on fitness
3. The population is evolved using mutation and crossover



Slide credit: Justin Carmichael and Sonny Trujillo Jr.

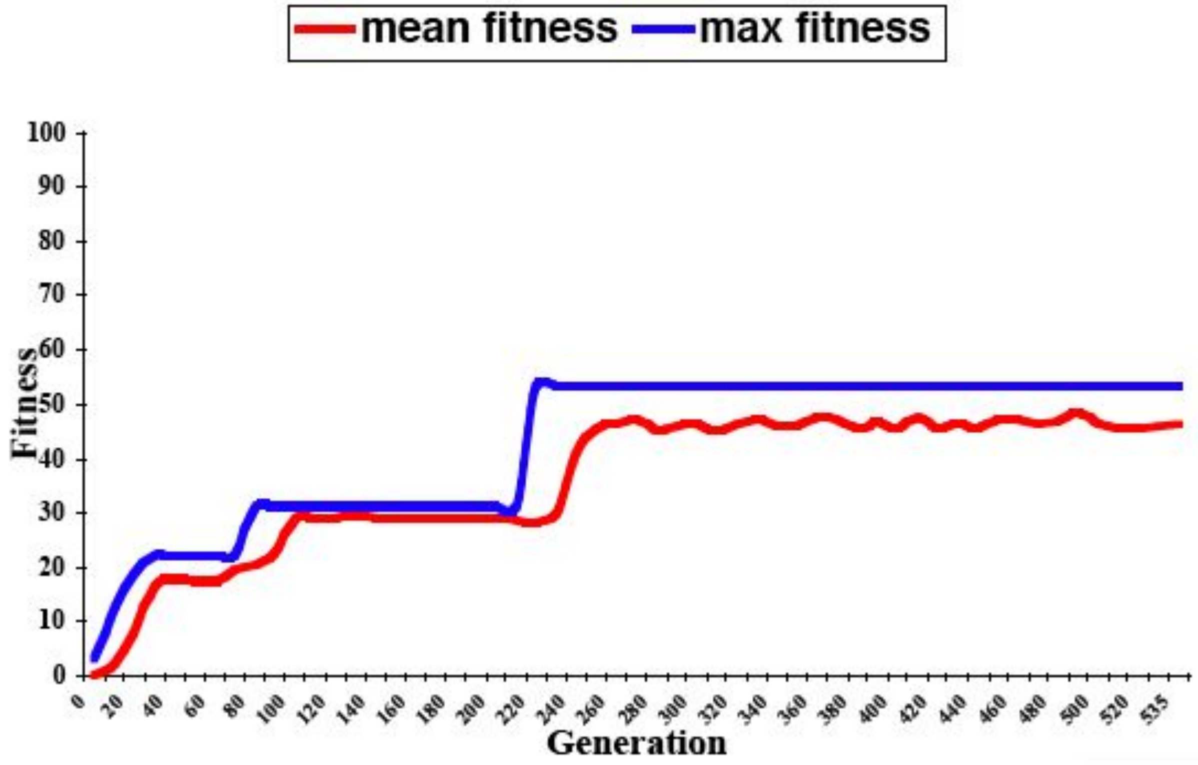
How it Works

1. A population of individuals is created in a computer.
2. Individuals are selected based on fitness
3. The population is evolved using mutation and crossover
4. The process repeats using the new population

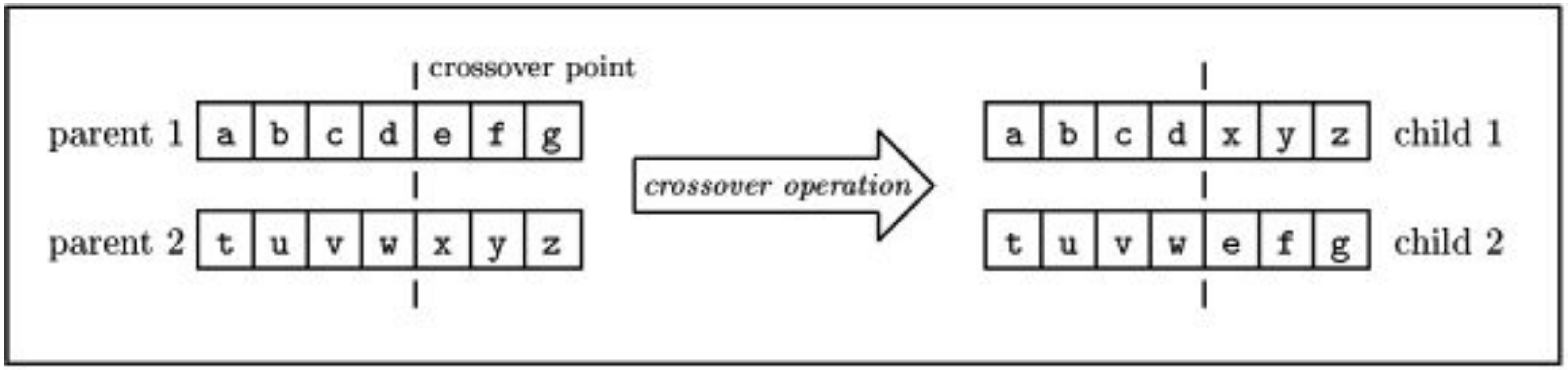


Slide credit: Justin Carmichael and Sonny Trujillo Jr.

Example Performance Curve



- Define the individuals
    - string of bits or letters, array of numbers, etc. (representation matters)
- Define a fitness function that evaluates the string (the function matters)
- Define the rules (rules matter)
    - selecting individuals (e.g. roulette or tournament)
    - mutation (e.g., some % of bits flip)
    - mating (usually 2 parents)
    - cross over (e.g., probability of crossing over at each position)
    - 1 point, 2 point, n point



- Fitness functions
  - Raw fitness: f = % of correct bits, selective pressure decreases as answer gets closer
  - Scaled fitness: 2^f One more correct letter is twice as fit (only works in simple cases)
  - Normalized fitness: fitness divided by the average fitness in the population

- Selection Methods
  - Fitness proportionate: roulette wheel—probability of appearing in P' is proportional to normalized fitness
  - Tournaments: pick (usually) 2 individuals from P, compare fitness, put more fit individual in P'. Sample with replacement.
  - Elitism—guarantee best x solutions will appear in P'
  - Implicit fitness in agent based models—ability to reproduce determines fitness

Simple example: evolve a string (Flake ch 20)

- Find the string “Furious green ideas sweat profusely”
- There are 27^{35} possible strings
- GA: 500 strings, crossover rate 75%, mutation rate 1%

Time, avg f, best f, Best string

0, .035, .20 “pjrmrubynrksiidwctxfodkodjjzfunpk”

1, .070, .26 “pjrmrubynrksxiidnybvswcqo piisyexdt”

26, .72, .80, “qurmous green idnasvsweqt prifuseky”

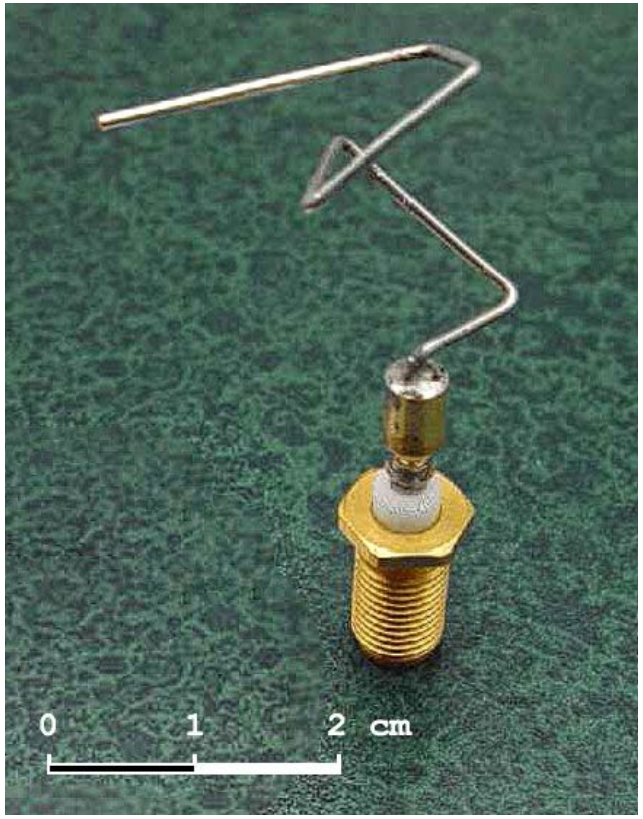
42, .90, .97, “qurious green ideas sweat profusely”

46 .94, 1, “curious green ideas sweat profusely”

Massively parallel directed search is effective when there is 1 correct answer.

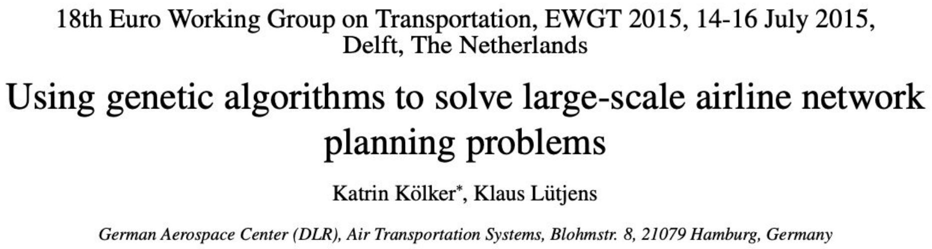
Flake CBoN

GAs: a useful meta-heuristic



The 2006 NASA ST5 spacecraft antenna. This complicated shape was found by an evolutionary computer design program to create the best radiation pattern. It is known as an evolved antenna.

https://en.wikipedia.org/wiki/Genetic_algorithm



18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands

Using genetic algorithms to solve large-scale airline network planning problems

Katrin Kölker*, Klaus Lütjens

German Aerospace Center (DLR), Air Transportation Systems, Blohmstr. 8, 21079 Hamburg, Germany

Genetic Programming

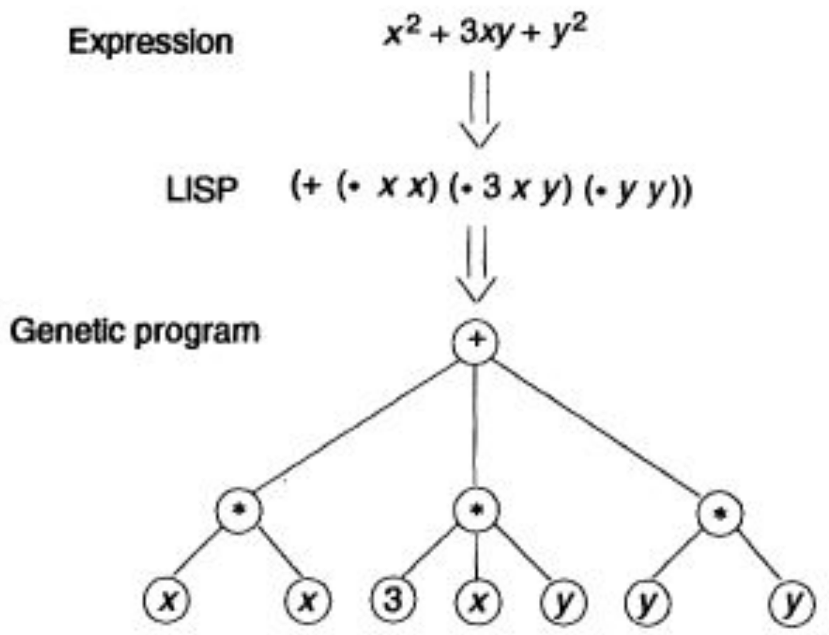


Fig. 4. Tree representation of computer programs. The displayed tree corresponds to the expression $x^2 + 3xy + y^2$. Operators for each expression are displayed as a root, and the operands for each expression are displayed as children.

cos (2x)

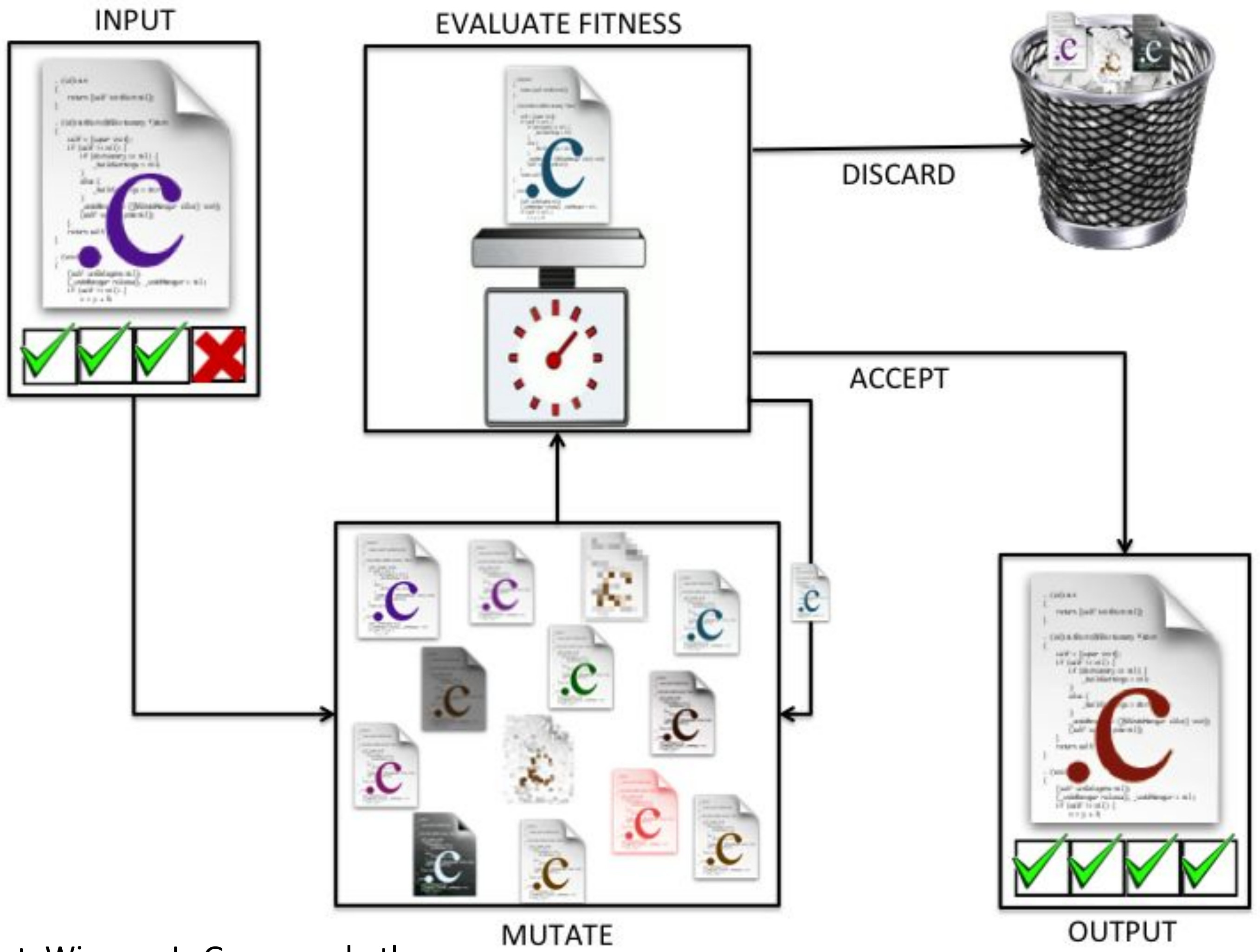
(−1 (* 2 (* sin (sin *x*)))), but genetic programming discovered (9, p. 241)

(sin (− (− 2 (* *x* 2))

(sin (sin (sin (sin (sin (sin (* (sin (sin 1))

(sin (sin 1)))))))))))

Genetic Programming



GenProg: Forrest, Wiemer, LeGoues and others

Example

- Initialize a population, P, of strings
- Define a fitness function that evaluates the string: Match mystery string _ _ _ _ _ _
- Define the rules for
- selecting individuals:
  - tournament (explain roulette)
  - with elitism (top string stays un-mutated)
- mutation: 15% (expect 1 of every 6 elements to mutate
  - (define mutate as random change))
- cross over 1 point cross over 75% of the time
- Repeat
  - Create an empty population, P’
  - Select 2 individuals from P based on fitness
  - Apply mutation, mating, crossover
  - Add the individuals to P’

Guidelines for implementing GAs

Define the individuals (string of bits or letters, representation matters)

Define a fitness function that evaluates the string (explicitly or implicitly, e.g. in an ABM)

Define the rules for
  selecting individuals (e.g. roulette or tournament, often with elitism)
  mutation (e.g., some % of bits flip)
  mating (usually 2 parents)
  cross over (e.g., probability of crossing over at each position;
    1 point, 2 point, n point)

Parameters (rough guidelines from DeJong 1975 GA experiments on a particular suite of problems):

– Bitstring length: 32 - 10,000
– Population size: 100 - 1000
– Length of run: 50 - 10,000
– Single point crossover rate: 0.6 per pair of parents
– Mutation rate: 0.005 per bit

Forrest, 1993

- Gray codes
- Ordering Problem for TSP (random key method)
- Genetic Programming
- Schema

Gray Codes

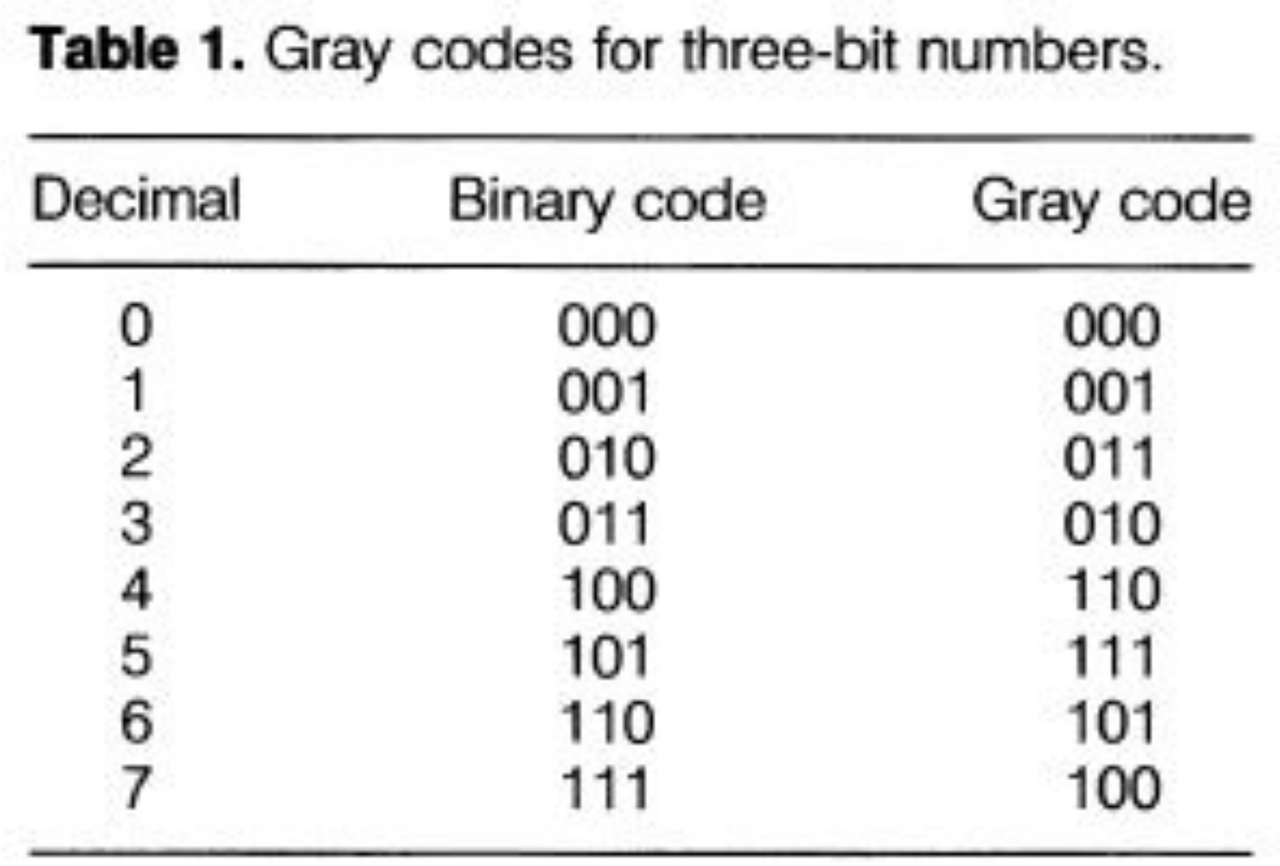
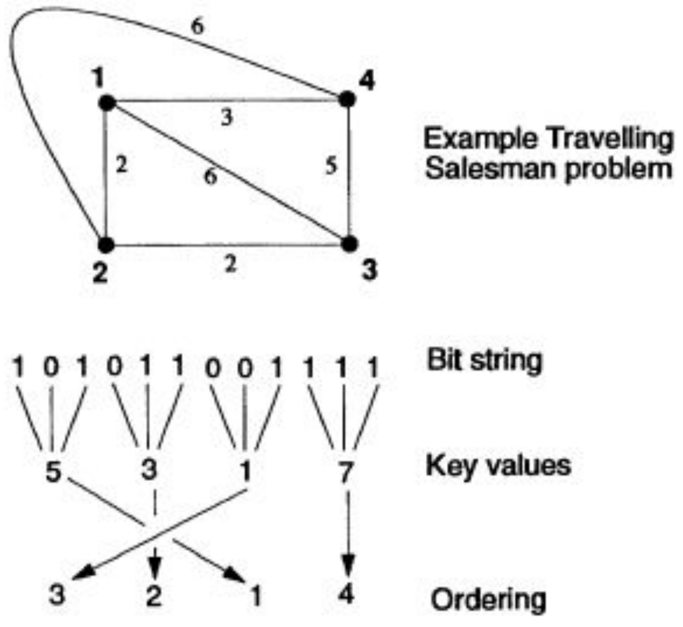


Table 1. Gray codes for three-bit numbers.

Decimal	Binary code	Gray code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

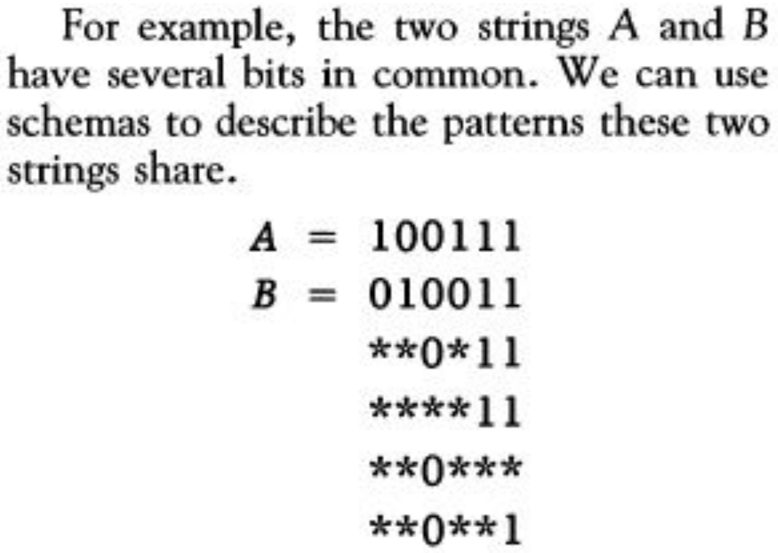
Ordering Problem for TSP (random key method)



$F(101011001111)$ = Tour Length(3,2,1,4) = 7

Fig. 3. The random-key representation for ordering problems. The example shows four cities (represented as nodes in the graph). Labels on the arcs denote the distance between cities.

Schema



For example, the two strings *A* and *B* have several bits in common. We can use schemas to describe the patterns these two strings share.

```
A = 100111
B = 010011
    **0*11
    ****11
    **0***
    **0**1
```

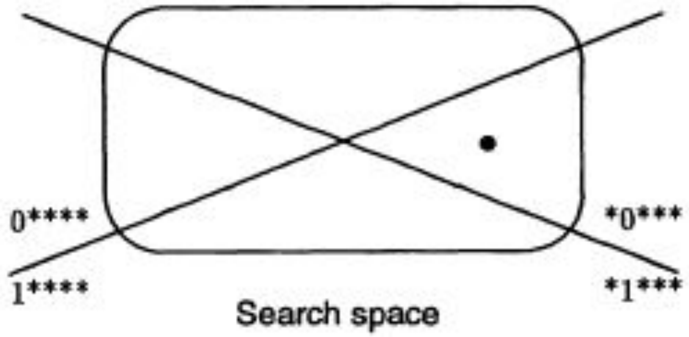


Fig. 5. Schemas define hyperplanes in the search space.