Genetic Algorithms

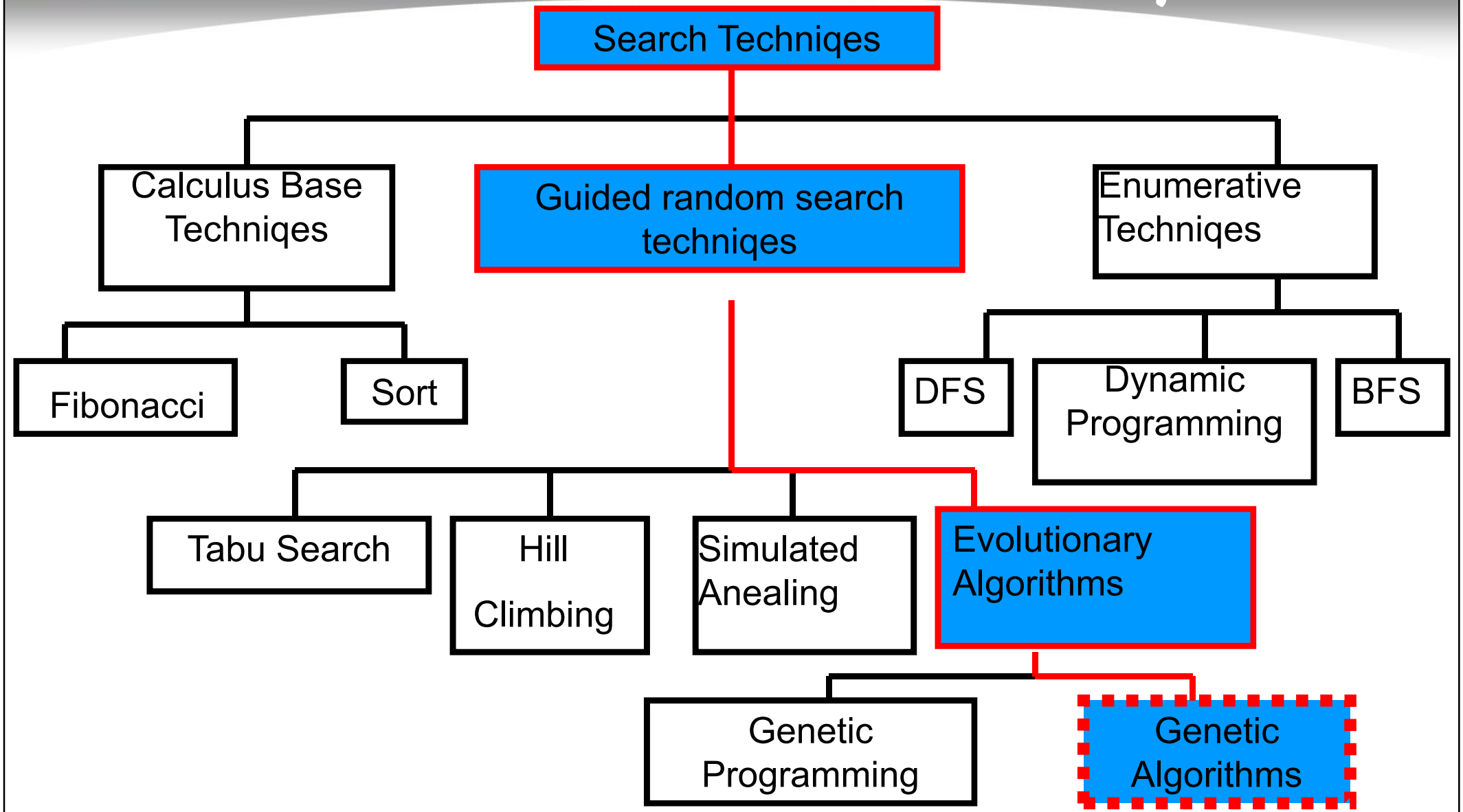
Prem Junsawang

Department of Statistics
Faculty of Science
Khon Kaen University

Overview

- Genetic Algorithms are ***search*** and ***optimization*** techniques based on Darwin's Principle of ***Natural Selection***
- GAs apply the same idea to problems where the solution can be expressed as an optimal individual and the goal is to maximize the fitness of individuals.
- The training proceeds in an evolutionary way, by having more fit individuals propagate their weights to succeeding generations. Less fit individuals do not survive.

Classes of Search Techniques



Why Genetic Algorithms?

- GAs can search spaces of patterns containing complex interaction parts, where the impact of each part on overall pattern fitness may be difficult to model.

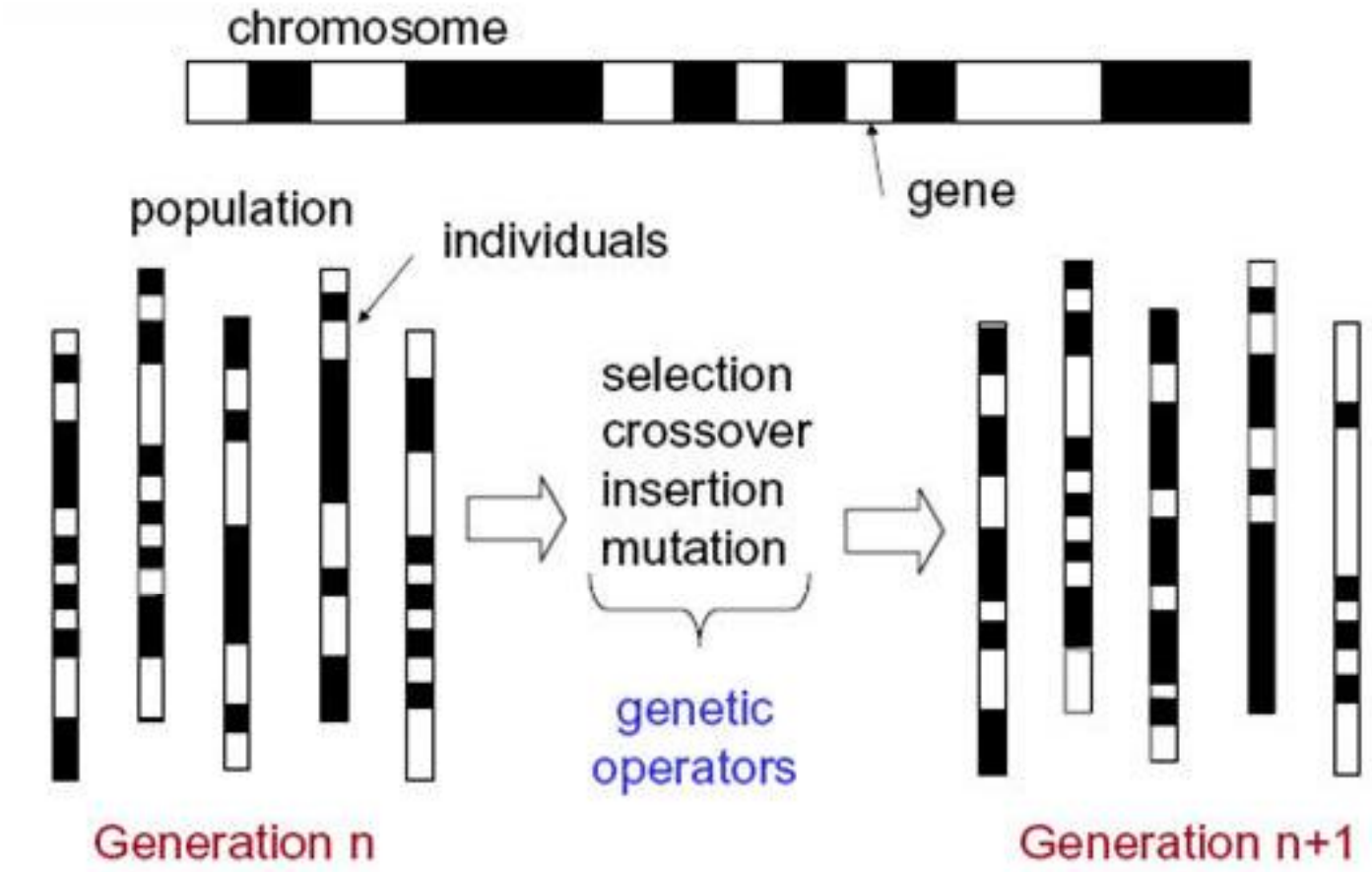
- GAs are easily parallelized and can take advantage of the decreasing computation costs.

How They Work(1)

- GAs work by evolving successive generations of genomes that progressively more and more fit(they provide better solutions to the problems).

How They Work(1)

GA Elements



Citation:

http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-888Spring-2004/D66C4396-90C8-49BE-BF4A-4EBE39CEAE6F/0/MSDO_L11_GA.pdf

How They Work(2)

- Evolution is simulated with the following steps:
  1. Identity the genome and fitness function
  2. Create the initial generation of genomes
  3. Modify the initial population by applying the operations of genetic algorithms
  4. Repeat Step 3 until the fitness of the population no longer improves

Representation (1)

- Patterns(members of the population) in GAs are often represented by bit strings, so that they can be manipulated by genetic operators.

- The bit string is composed of a set of specific substring for each rule precondition (condition of attributes) and postcondition (class value).

Representation (2)

- If we concatenate substrings of attributes and class, we will get a bit string of a particular pattern and we can represent sets of rules by similarly concatenating the bit string representation of the individual rules.

Example 1

- For example, attribute *Outlook* can be represented a three-bit string.

Outlook	Representation
Sunny	100
Overcast	010
Rain	001
Sunny v Rain	101

Example 2

- Same with class values.

Play Tennis	Representation
Yes	10
No	01
Or Yes	1
Or No	0

Note 1

- The bit string representing rule contains a substring for each attribute is not included in the rule precondition.
- For example,

  If *wind* = *strong* Then *PlayTennis* = *Yes*

Outlook	Wind	PlayTennis
111	10	01

Note 2

- An entire rule can be described by concatenating the bit strings describing the rule pre- and post-conditions.
- This yields a fixed length bit-string in which sub-string at a specific locations describe specific attributes.

Precedure (1)

1. Generate random population of *n* chromosomes.
2. Evaluate the fitness of each chromosome in the population
3. Create new population by the following steps
   a) Select two parent chromosomes from the population according to their fitness
   b) Crossover the patents to form new offsprings
   c) Mutation of each offspring

Precedure (2)

4. Use generated population for further process

   Steps 2-4

5. If the condition is satisfied

   stop and return the best solution in the current population

   else

   go to step 2

   end

Genetic Operator

- Selection
- Crossover
- Mutation

Genetic Operator: Selection1

- Only the fitness individuals in the population survive to pass their genetic material on to the next generation.
- However, the size of the population remains constant from one generation to the next.
- The chance of a genome surviving to the next generation is proportional to its fitness value.

Genetic Operator: Selection2

- Rank selection
  - The patterns are first sorted by fitness and they are selected based on their rank in the sorted list, rather than fitness.
- The tournament selection
  - Two patterns are first chosen at random from the current population and the more fit of these two is selected

Genetic Operator: Selection3

- Fitness proportionate selection
  - The ratio of the fitness of a particular pattern to the fitness of other members of the current population (Roulette wheel selection)

Genetic Operator: Selection3

- Fitness proportionate selection
  - The ratio of the fitness of a particular pattern to the fitness of other members of the current population (Roulette wheel selection)

Genetic Operator: Selection3

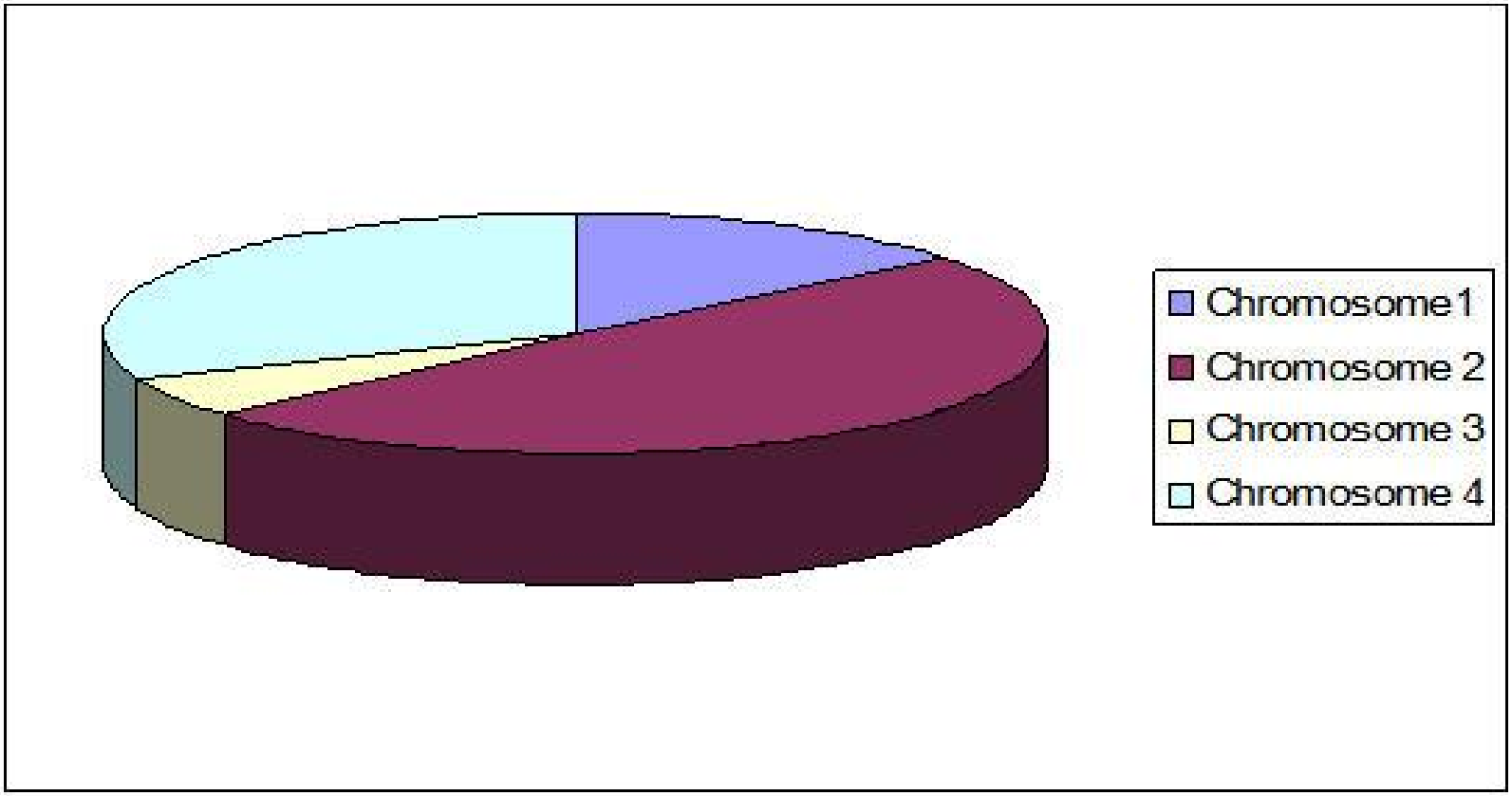
Example of Roulette wheel selection

No.	String	Fitness	% Of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

Citation: : www.cs.vu.nl/~gusz/

Genetic Operator: Selection3

Roulette wheel selection



All you have to do is spin the ball and grab the chromosome at the point it stops ☺

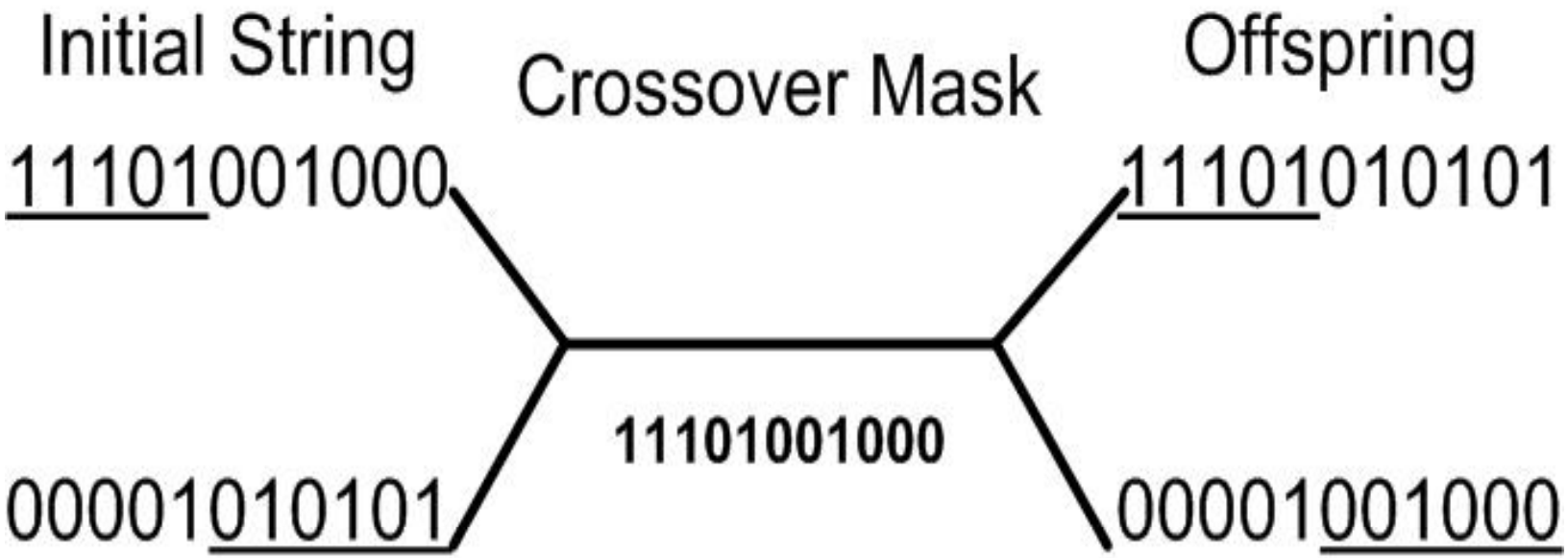
Genetic Operator: Crossover1

- Crossover operator produces two new offspring from two parents string, by copying selected bits from each parent.

- The choice of which parent contributes the bit position is determined by a string called the *crossover mask*.

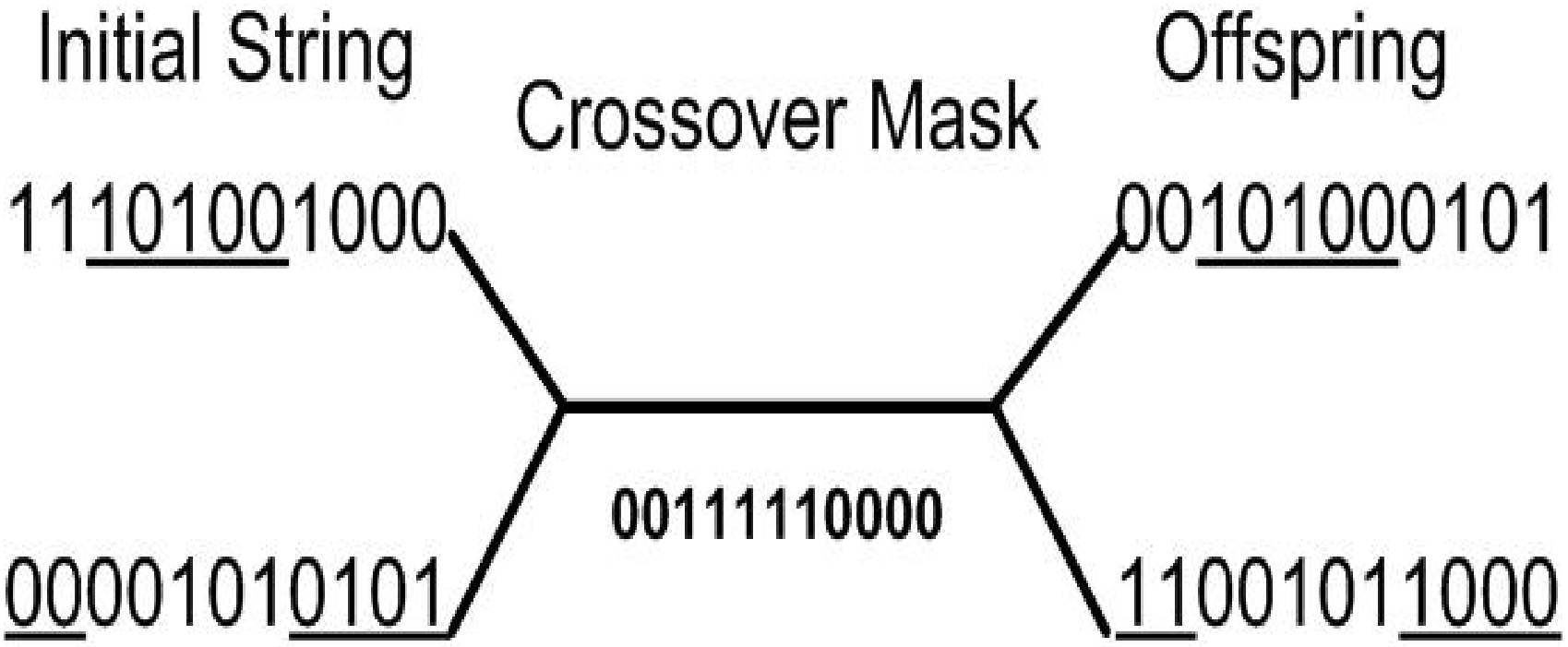
Genetic Operator: Crossover2

- Single-point crossover



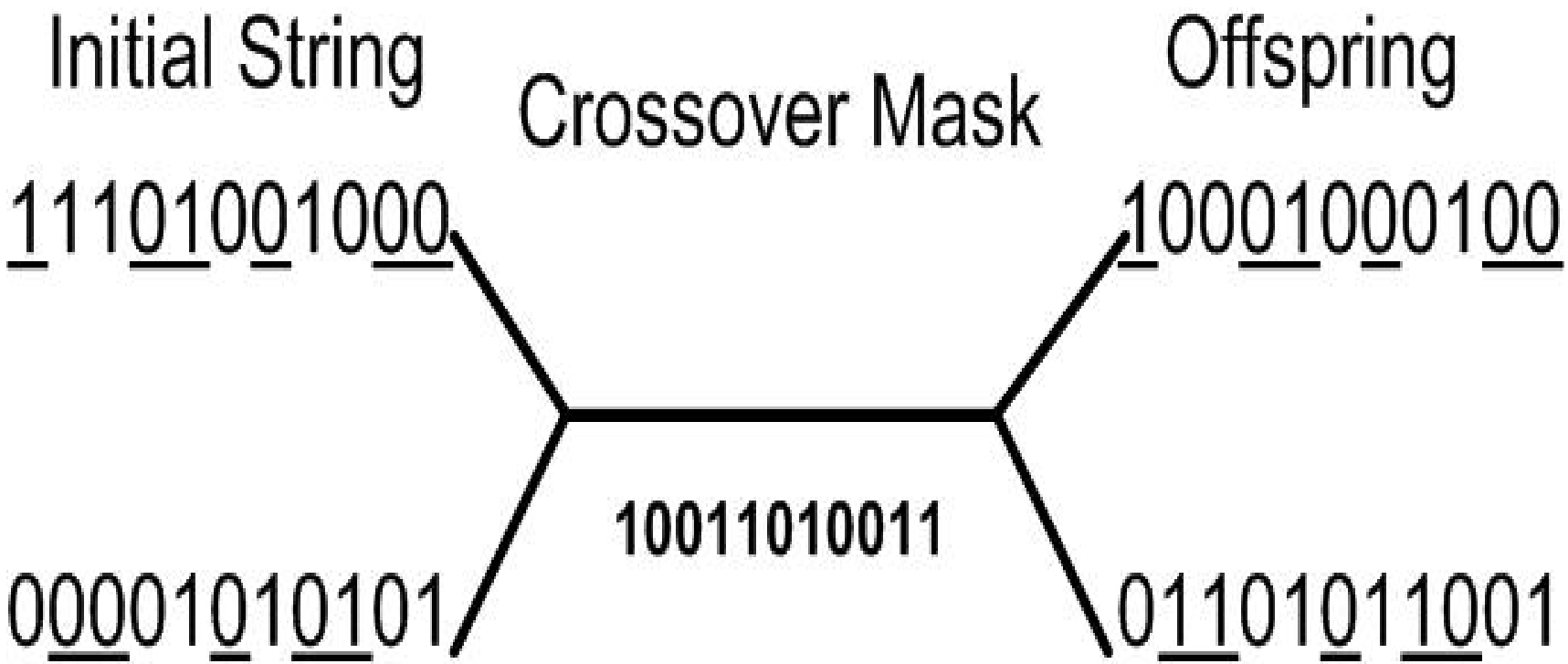
Genetic Operator: Crossover3

- Two-point crossover



Genetic Operator: Crossover4

- Uniform crossover



Genetic Operator: Mutation1

- The mutation operator produces small changes to the bit string by choosing a single bit at random, then changing its value.
- Mutation is often performed after crossover has been applied.
- Selection and crossover do a good job of searching the space of possible genomes.

Genetic Operator: Mutation2

- The mutation rate is quite small in nature and is usually kept quite low for genetic algorithms.

Fitness Functions

- The criterion for ranking potential patterns for probabilistically selecting them for inclusion in the next generation population eg, accuracy, complexity of the rules

Input Parameters

- The fitness functions for ranking candidate patterns(eg. The accuracy of the patterns over training data).
- A threshold for defining an acceptable level of fitness for terminating the algorithm.
- The size of population (*p*) to be maintained
- The fraction of population (*r*) to be replaced at each generation
- The mutation rate(*m*)

Algorithm1

Step1: *p* patterns are randomly generated

Step2: Each pattern is evaluated by the given fitness function

Step3: While stopping criterion is not met create a new population by the following

1. select (1-*r*)*p* best pattern from the current population
2. Select (r*p)/2 from 1 and apply crossover operator to produce two offspring from each pair of patterns.

Algorithm1

3. Randomly apply *mutation* operation to *m*% of the current population
4. Evaluate each pattern by the fitness function.

Benefits

- Always an answer: answer get better with time.
- Good for ‘noisy’ environments,
- Inherently parallel: easily distributed

Drawback

- There are many parameters involving GA, such as size of population, crossover mask, mutation rate, fraction of population to be replaced, it is not clear what is the ideal parameter setting for each problem domain.
- GA involves considerable computation and overhead, they tend to be used with relatively small data sets.

Example: MaxOne Problem

Suppose we want to maximize the number of ones in a string of l binary digits

It may seem so because we know the answer in advance

However, we can think of it as maximizing the number of correct answers, each encoded by 1, to l yes/no difficult questions

Example: MaxOne Problem

- The fitness f of a candidate solution to the MAXONE problem is the number of ones in its genetic code
- We start with a population of n random strings. Suppose that $l = 10$ and $n = 6$

Example: MaxOne Problem

We toss a fair coin 60 times and get the following initial population:

s_1 = 1111010101 $f(s_1) = 7$

s_2 = 0111000101 $f(s_2) = 5$

s_3 = 1110110101 $f(s_3) = 7$

s_4 = 0100010011 $f(s_4) = 4$

s_5 = 1110111101 $f(s_5) = 8$

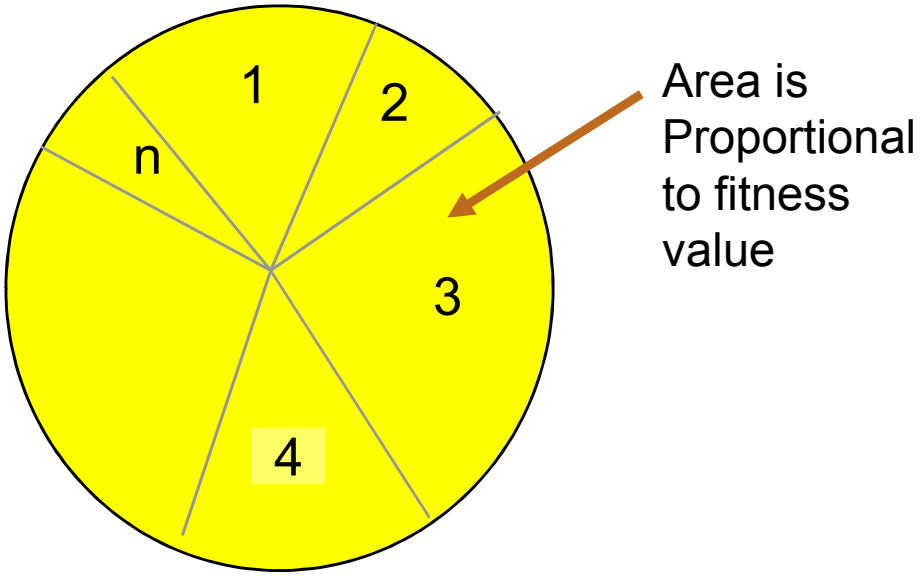
s_6 = 0100110000 $f(s_6) = 3$

Example: MaxOne Problem

Next we apply fitness proportionate selection with the roulette wheel method:

Individual *i* will have a probability to be chosen $\frac{f(i)}{\sum_i f(i)}$

We repeat the extraction as many times as the number of individuals we need to have the same parent population size (6 in our case)



Example: MaxOne Problem

Suppose that, after performing selection, we get the following population:

s_1` = 1111010101 (s_1)

s_2` = 1110110101 (s_3)

s_3` = 1110111101 (s_5)

s_4` = 0111000101 (s_2)

s_5` = 0100010011 (s_4)

s_6` = 1110111101 (s_5)

Example: MaxOne Problem

Next we mate strings for crossover. For each couple we decide according to crossover probability (for instance 0.6) whether to actually perform crossover or not

Suppose that we decide to actually perform crossover only for couples (s_1`, $s_2$`) and (s_5`, $s_6$`). For each couple, we randomly extract a crossover point, for instance 2 for the first and 5 for the second

Example: MaxOne Problem

Before crossover:

s_1` = 11111010101

s_2` = 11110110101

s_5` = 01000010011

s_6` = 11101111101

After crossover:

s_1`` = 11110110101

s_2`` = 11111010101

s_5`` = 01000011101

s_6`` = 11101110011

Example: MaxOne Problem

The final step is to apply random mutation: for each bit that we are to copy to the new population we allow a small probability of error (for instance 0.1)

Before applying mutation:

s_1`` = 11101**1**0101

s_2`` = 1111**0**10101

s_3`` = 11101111**0**1

s_4`` = 0111000101

s_5`` = 0100011101

s_6`` = 1110110011

Example: MaxOne Problem

After applying mutation:

$s_1''' = 11101\mathbf{0}0101$ $\quad f(s_1''') = 6$

$s_2''' = 1111\mathbf{1}10\mathbf{1}0\mathbf{0}$ $\quad f(s_2''') = 7$

$s_3''' = 11101\mathbf{0}111\mathbf{1}$ $\quad f(s_3''') = 8$

$s_4''' = 0111000101$ $\quad f(s_4''') = 5$

$s_5''' = 0100011101$ $\quad f(s_5''') = 5$

$s_6''' = 1110110\mathbf{0}\mathbf{1}$ $\quad f(s_6''') = 6$

Example: MaxOne Problem

In one generation, the total population fitness changed from 34 to 37, thus improved by ~9%

At this point, we go through the same process all over again, until a stopping criterion is met

Case Study: Business Context

- The customer service department of a major international airline processes many customer comments which arrive via several channels.
- Different comment have different priorities for response.

Case Study: Business Context

- Airline personnel spend significant amounts of time analyzing customer comments.
- This airline decided to reduce the time it took to respond to a complaint by automating the initial categorization of comments.

Case Study: Data

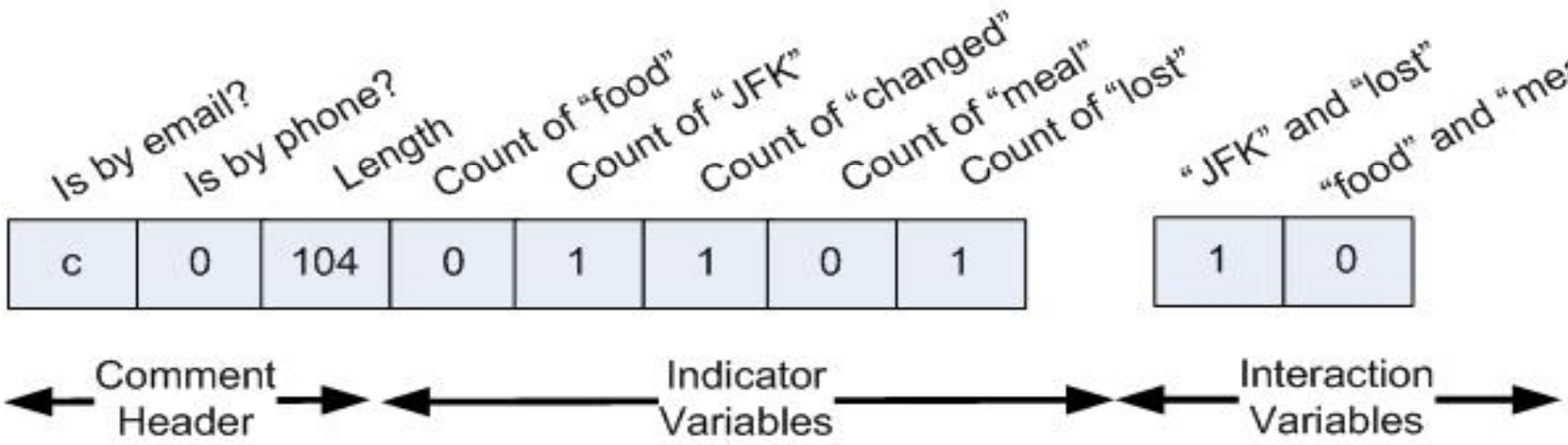
- A complete customer comment record has the following fields:
  - Date
  - Source (eg. Email, telephone contact)
  - Flight number
  - Class of service
  - Departure airport
  - Destination airport
  - Mileage account number
  - Free-text comments

Case Study: Evolving a Solution

- The comment signature describes the text in the comment.

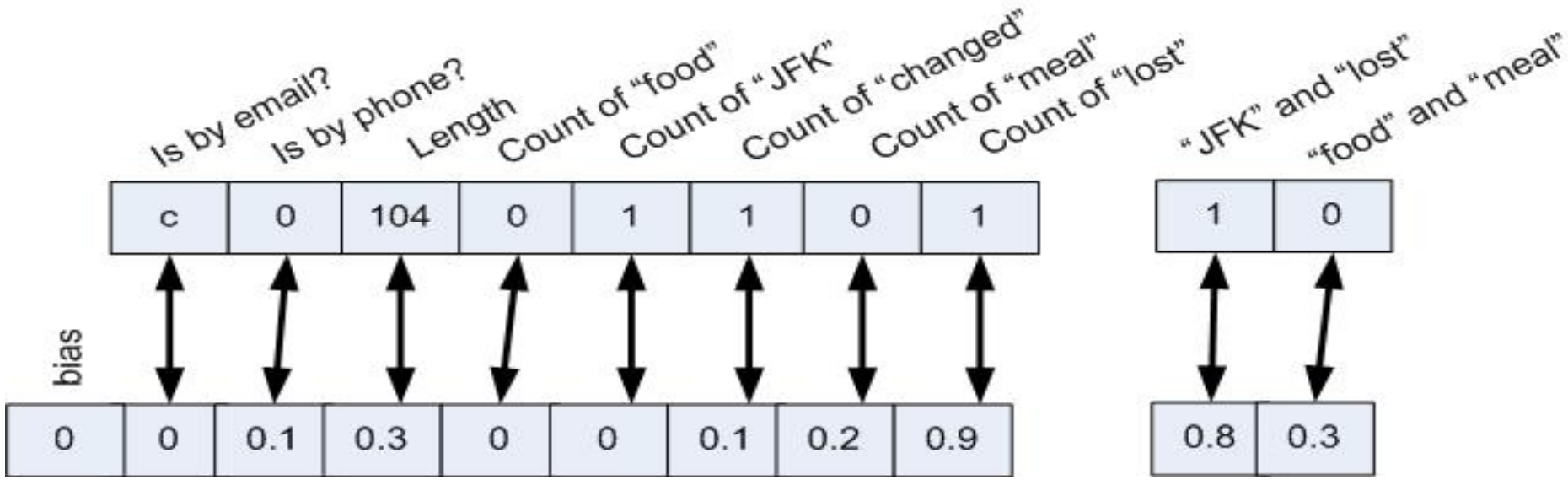
To : Comments@airline.com
From: random_customer

… My baggage was lost at JFK when I changed planeds …



Case Study: Evolving a Solution

- The comment signature is not the genome but is related to it.
- The genome is a set of weights corresponding to each variable in the signature



Summary

- The success of applying genetic algorithms depends on two things.
  - The genome
  - The fitness function
- The genome encodes the problem.
- The fitness function makes sense of the genome.
- Genetic algorithms work on a wide variety of fitness functions, making it possible to encode many different types of problems that are not easily handled by other means.
- The process of evolution starts with a random population and then applies three transformation operators.

Summary

- Selection operator makes more fit genomes survive from one generation to another.
- Crossover enables genomes to swap pieces.
- Mutation enables some values to change randomly.
- The application of these three operators produce a new generation whose average fitness should be greater than original.

References

- P. N. Tan, M Steinbach, V. Kumar,"Introduction to data mining", Pearson Addison Wesley.
- เอกสารประกอบการสอนรายวิชา KNOWLEDGE / DATA MINING โดย ผศ.ดร. จันทร์เจ้า มงคลนาวิน