Evolutionary Computation

(GENETIC ALGORITHM)

Dr. ZAKARIA ABDULWAHID HAMED

Adopted From Lecture by Michael Negnevitsky , Electrical Engineering & Computer Science University of Tasmania and Lecture by prof. Dr. Siti Maryam and Prof. Dr. Sitizaiton, FC, UTM

The Objectives

- To understand the processes i.e. GAs Basic flows-operator, parameters involved (roles, effects etc)
- To be able to know how to apply Ga s in solving optimization problems.

Outline

Introduction: Evolutionary computation

- Evolutionary computation simulates evolution on a computer. There result of such a simulation is a series of optimization algorithms, usually based on a simple set of rules.

Introduction: continue

- The behavior of an individual organism is an inductive inference about some yet unknown aspects of its environment. If, over successive generations , the organism survives, we can say that this organism is capable of learning to predict changes in its environment

- The evolutionary approach is based on computational models of natural selection and genetics. We call the evolutionary computation, an umbrella term that combines genetic algorithms and evolutionary strategies.

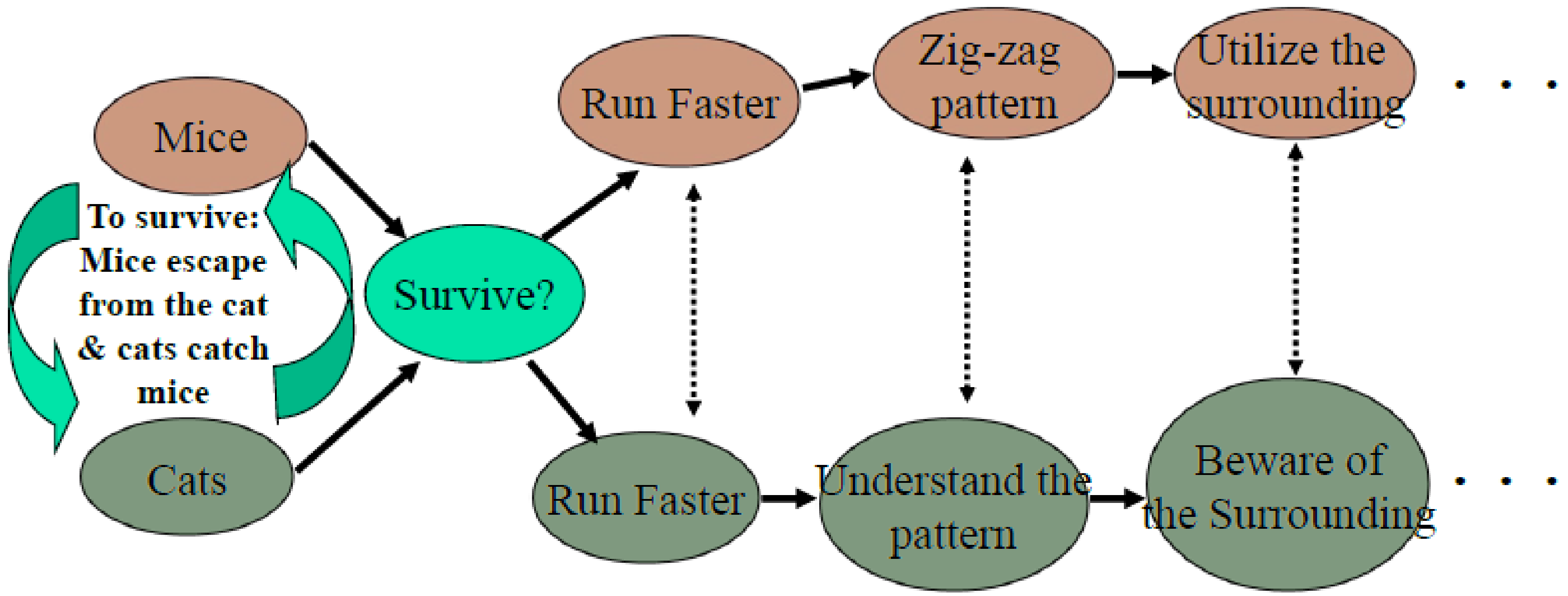
Simulation of Natural Evolution

- Natural Evolution .. process of reproduction, mutation, competition and selection.
- The power to reproduce appears to be an essential property of life.
- The power to mutate is also guaranteed in any living organism that reproduces itself in a continuously changing environment.
- Processes of competition and selection normally take place in the natural world, where expanding populations of different species are limited by a finite space.

Continue

- Evolution can be seen as a process leading to the maintenance of a population’s ability to survive and reproduce in a specific environment. This ability is called evolutionary fitness.
- Evolutionary fitness can also be viewed as a measure of organism’s ability to anticipate changes in its environment.
- The better an organism’s fitness to the environment, the better its chances to survive

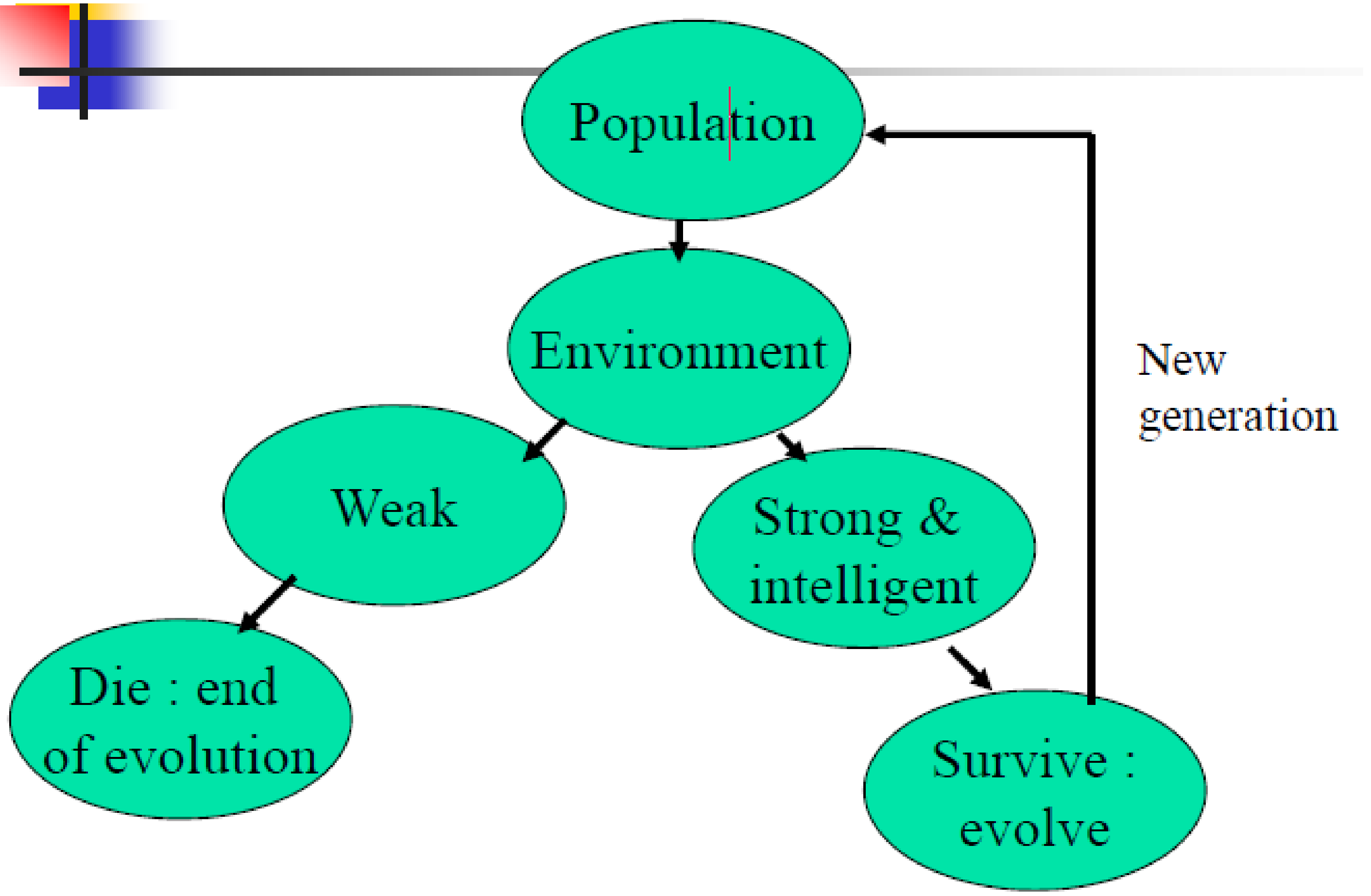
mice & cats : an evolutionary process



The Mice & Cats algorithm

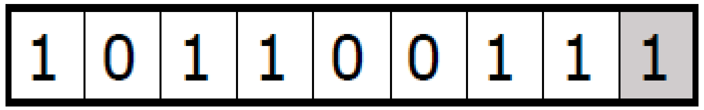
- Mice & Cats want to survive .
- While (mice != dead || cats != dead)
- if (speed(mice) > speed(cats))
- mice = survive else cats = survive
- if (pattern(mice) > pattern(cats)
- mice = survive else cats = survive
- if (surrounding(mice) > surrounding(cats)
- mice = survive else cats = survive
- *
- *
- *
- endofwhile

General Evolutionary Process



Genetic Algorithm

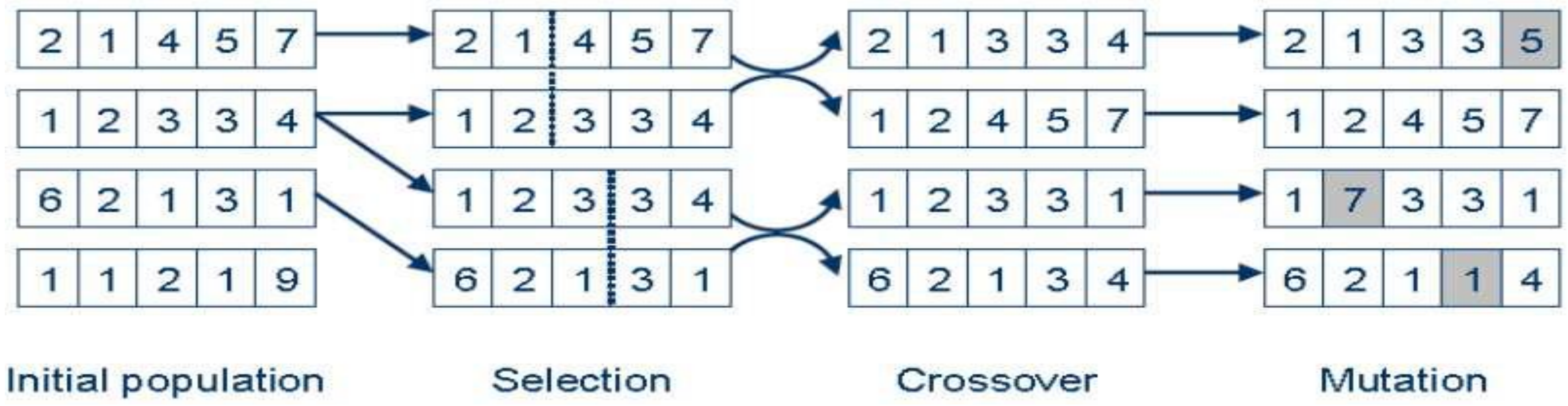
- In the early 1970’s John Holland introduced the concept of genetic algorithms. His aim was to make computer do what nature does. Holland was concerned with algorithms that manipulate strings of binary digits.
- Genetic Algorithm is a type of local search that mimics evolution by taking a population of strings which encode possible solutions and combines them based on a fitness function to produce individuals that are more fit.
- Genetic Algorithms are computer programs that evolve in ways that look like natural selection and can be applied to solve complex problems.
- Genetic algorithms are a type of optimization algorithm, meaning they are used to find the optimal solution(s) to a given computational problem that maximizes or minimizes a particular function.
- Artificial “chromosomes” consist of a number of “genes” , and each gene is represented by 0 or 1



Genetic Algorithm Cont.

- A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems. GAs are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).
- The evolution usually starts from a population of randomly generated individuals and happens in generations.
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness) and modified to form a new population. The new population is used in the next iteration of the algorithm
- The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Example



Individual - Any possible solution
Population - Group of all individuals
Fitness – Target function that we are optimizing (each individual has a fitness)

The basic components common to almost all genetic algorithms are:

- A fitness function for optimization
- A population of chromosomes
- Selection of which chromosomes will reproduce
- Crossover to produce next generation of chromosomes
- Random mutation of chromosomes in new generation

- The fitness function is the function that the algorithm is trying to optimize.
- The term chromosome refers to a numerical value or values that represent a candidate solution to the problem that the genetic algorithm is trying to solve.
- A genetic algorithm begins with a randomly chosen group of chromosomes, which serves as the first generation (initial population). Then each chromosome in the population is evaluated by the fitness function to test how well it solves the problem at hand.

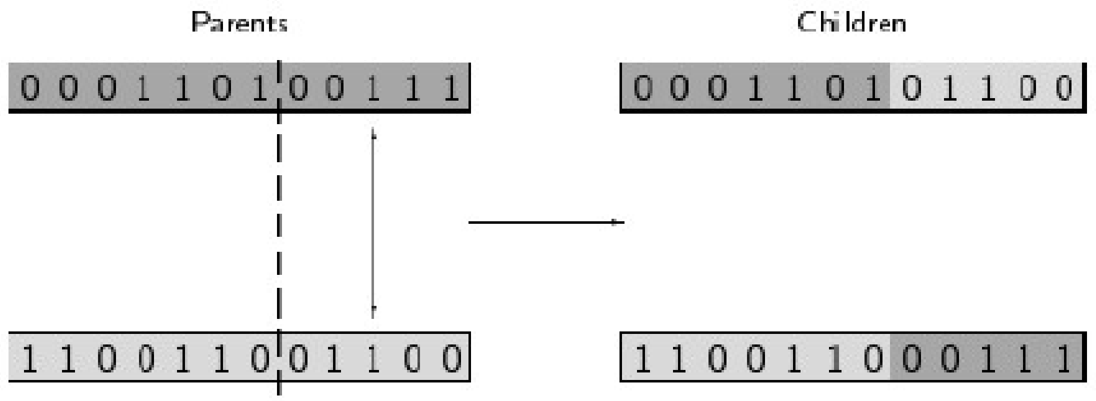
Stochastic operators

- The **selection operator** chooses some of the chromosomes for reproduction based on a probability distribution defined by the user. The fitter a chromosome is, the more likely it is to be selected. For example, if f is a non-negative fitness function, then the probability that chromosome $f(x)$ is chosen to reproduce might be:

$$P = \frac{f(x)}{\Sigma f(x)}$$

Crossover and mutation

- The crossover operator resembles the biological crossing over and recombination of chromosomes in cell divisions. This operator swaps a subsequence of two of the chosen chromosomes to create two offspring
- Basically, crossover is the exchange of genes between the chromosomes of the two parents. In the simplest case, we can realize this process by cutting two strings at a randomly chosen position and swapping the two tails. This process, which we will call one-point crossover in the following, is visualized in Figure below.



Mutation

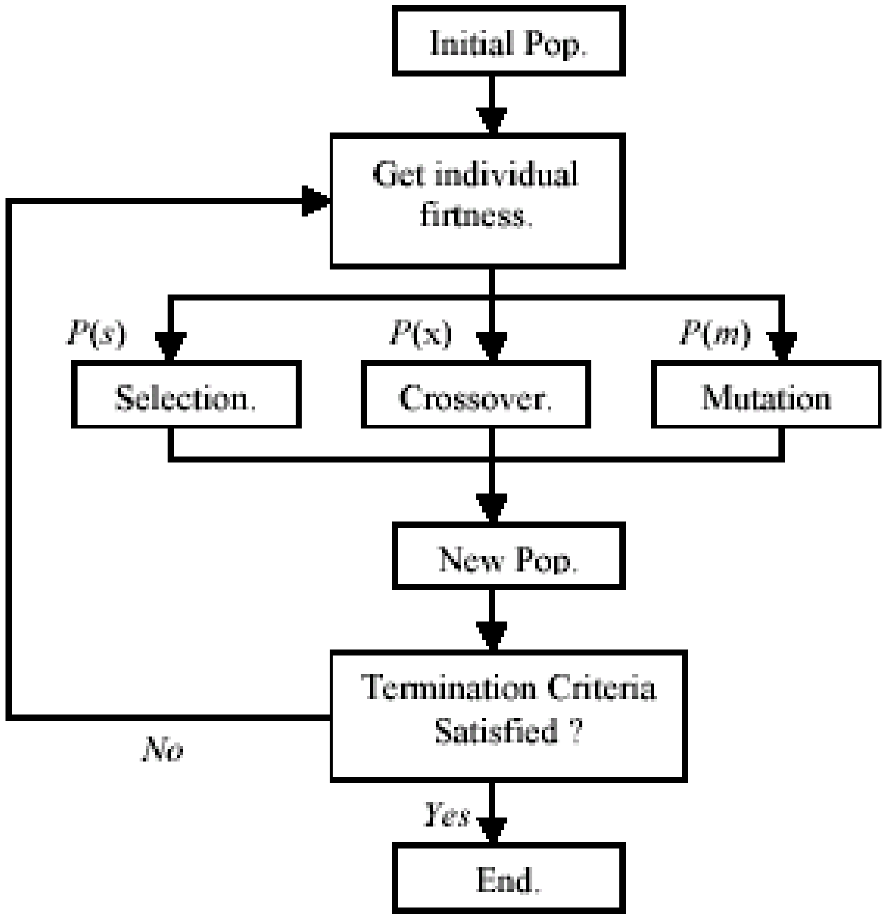
- The mutation operator randomly flips individual bits in the new chromosomes (turning a 0 into a 1 and vice versa).
- In real reproduction, the probability that a certain gene is mutated is almost equal for all genes

Basic Genetic Algorithm

- Step1:Represent the problem variable domain as a chromosomes of a fixed length. Choose size of population N, probability of mutation p_m and crossover p_c
- Step2:Define fitness function to measure performance of individual chromosomes.
- Step3:Randomly generate initial population of chromosomes of sized N.
- Step4:Calculate the fitness of each individual chromosomes.

- Step5:Select a pair of “fit” chromosomes for mating.
- Step 6: Create a pair of offspring chromosomes by applying crossover and mutation.
- Step 7: Place the created offspring chromosomes in the new population.
- Step 8: repeat Step 5until the size of the new population equal to size of initial population, N
- Step 9: Replace the initial (parent) chromosomes with the new (offspring) population.
- Step 10: Go to Step 4and repeat the process until the termination criterion satisfy.

The flowchart of GA algorithm



How Roulette Wheel Selection Works

- Calculate the total fitness of the population.
- Calculate the relative fitness for each individual by dividing their fitness by the total fitness.
- Calculate the cumulative probability for each individual by summing up the relative finesses.
- Generate a random number between 0 and 1.
- Select the first individual whose cumulative probability is greater than or equal to the random number.

Let's go through an example to understand this better.

Example

- Suppose we have a population of four individuals with the following fitness scores
- Individual A: 12
- Individual B: 8
- Individual C: 6
- Individual D: 4

1- Calculate the total fitness: Total Fitness = 12 + 8 + 6 + 4 = 30

2- Calculate the relative fitness:

Individual A: 12 / 30 = 0.4

Individual B: 8 / 30 = 0.2667

Individual C: 6 / 30 = 0.2

Individual D: 4 / 30 = 0.1333

3- Calculate the cumulative probability:

Individual A: 0.4

Individual B: 0.4 + 0.2667 = 0.6667

Individual C: 0.6667 + 0.2 = 0.8667

Individual D: 0.8667 + 0.1333 = 1

4- Generate a random number between 0 and 1:

Random number = 0.52

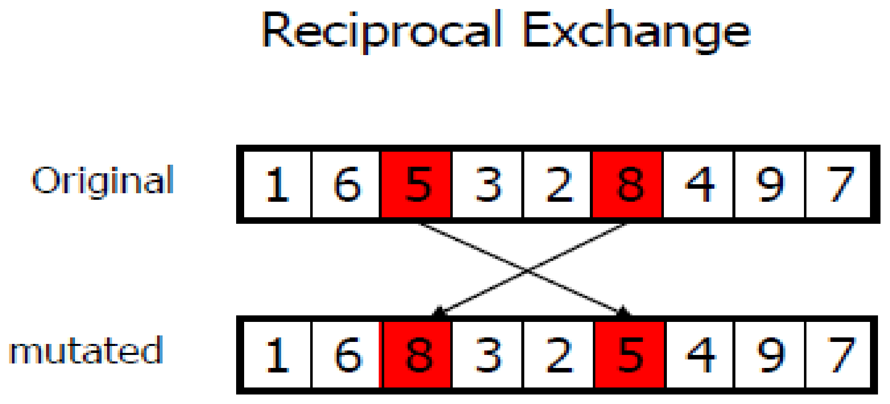
5- Select the parent:

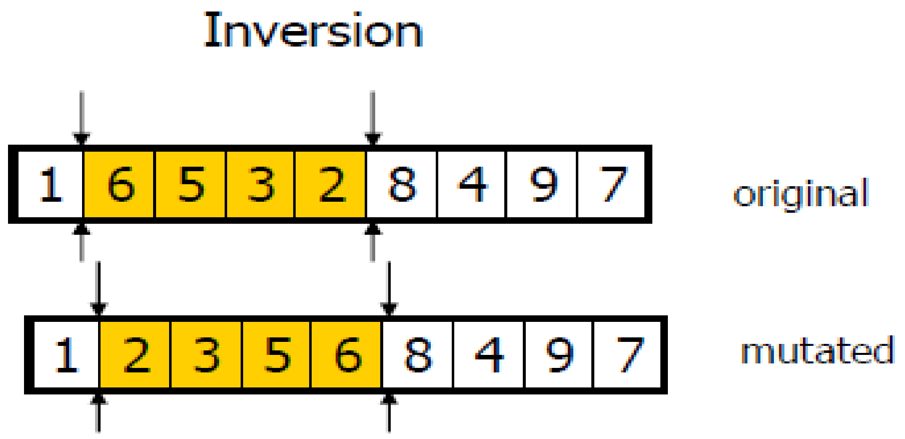
In this case, the random number falls between the cumulative probabilities of Individual A (0.4) and Individual B (0.6667). So, Individual B is selected as a parent, Repeat the process to select another parent. Then perform crossover and mutation to create offspring

Types of mutation

- Bit-Flip Mutation (inversion)
- In a bit-flip mutation, we randomly choose bits to flip (0 to 1 or 1 to 0):
- Swap Mutation

Swap mutation randomly selects two genes and swaps them:





Example 1: Burger and Profit Problem

- The goal :To help the owner of a restaurant find the best combination of possible solutions (we will see in next slide) that produce the highest profit.

Decision To make ; which is more profitable??

- Price: Should the price of the burger be RM1 or RM2?
- Drink : Should coke or pepsi be served with the burger ?
- Speed of Service: Should the restaurant provide slow, leisurely service by waiter with tuxedos or fast, snappy service by waiters in white polyester uniform ?

The solution:

- There are 3 decision variables each of which can assume one of two possible values.
- Assuming L is the length of string of possible strategy or combination so L = 3.
- Assuming K is the possible value for the decision variable so K = 2.
- The search space for this problem is 2^3= 8 possible strategies but we consider only 4 possible combinations.

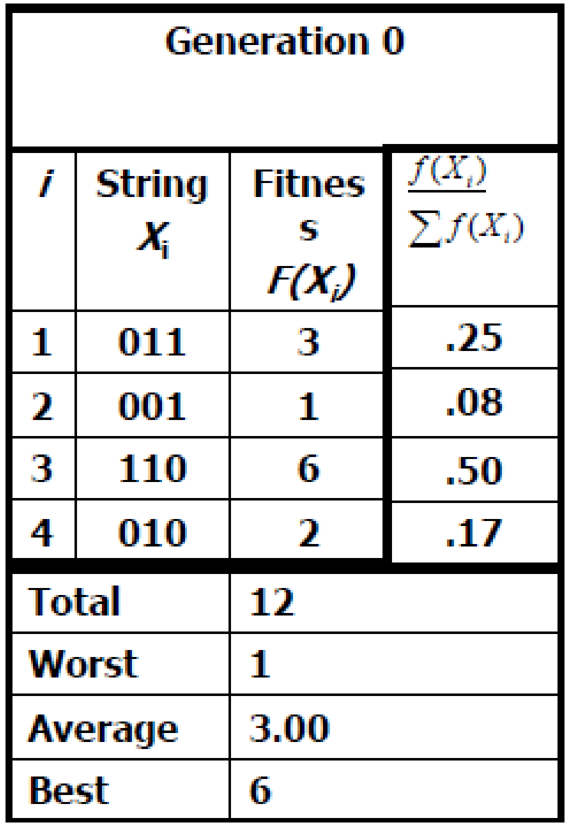
Problem representation

- We can represent the problem (decision to make) in binary. As shown below:
- Price (0 represents RM1; 1 represents RM2)
- Drink (0 represents Pepsi; 1 represents Cola)
- Speed (0 represents Leisurely; 1 represents Fast)

Strategy	Price	Drink	Speed	Binary
1	RM1	Cola	Fast	011
2	RM1	Pepsi	Fast	001
3	RM2	Cola	Leisurely	110
4	RM1	Cola	Leisurely	010

fitness function

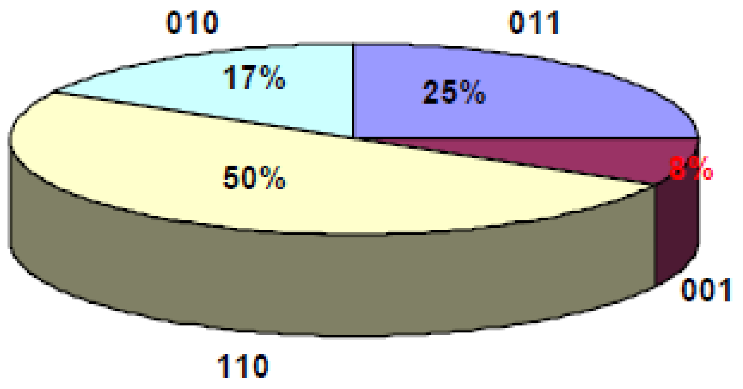
- Fitness = Profit
- For Simplicity .. The fitness is evaluated based on the binary value



Generation 0

i	String X_i	Fitness $F(X_i)$	$\frac{f(X_i)}{\sum f(X_i)}$
1	011	3	.25
2	001	1	.08
3	110	6	.50
4	010	2	.17

Total	12
Worst	1
Average	3.00
Best	6



Selection/Reproduction

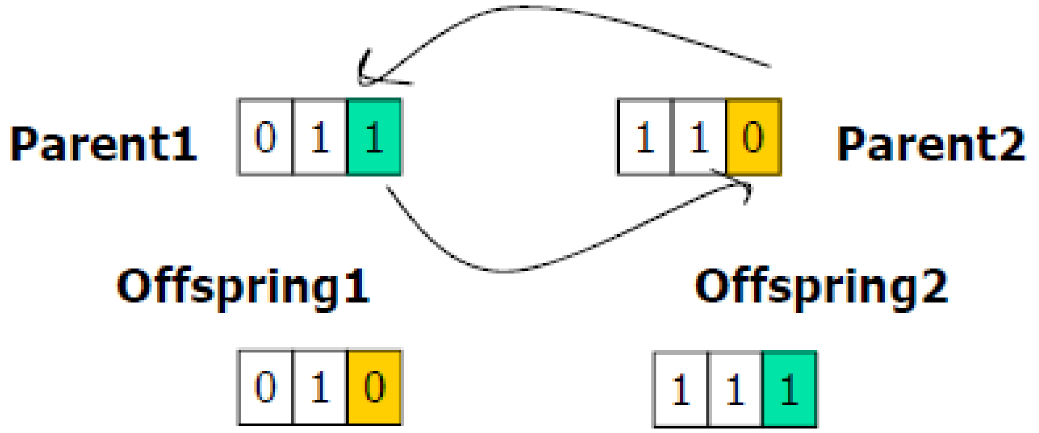
- Roulette Wheel Approach110 has possibility of 0.50 to reproduce. In theory, it will be selected 2 x in next generation.
- 011 has possibility of 0.25 ie. ¼ chances to reproduce.
- The same goes to the other string/ individual.
- BUT since GA is probabilistic, that the string 110 can be chosen 3X or even none in the next gen. The same applies to others

Mating Pool Selected for Reproduction	
Mating Pool	Pool $F(X_i)$
011	3
110	6
110	6
010	2
Total	17
Worst	2
Average	4.25
Best	6

Xover

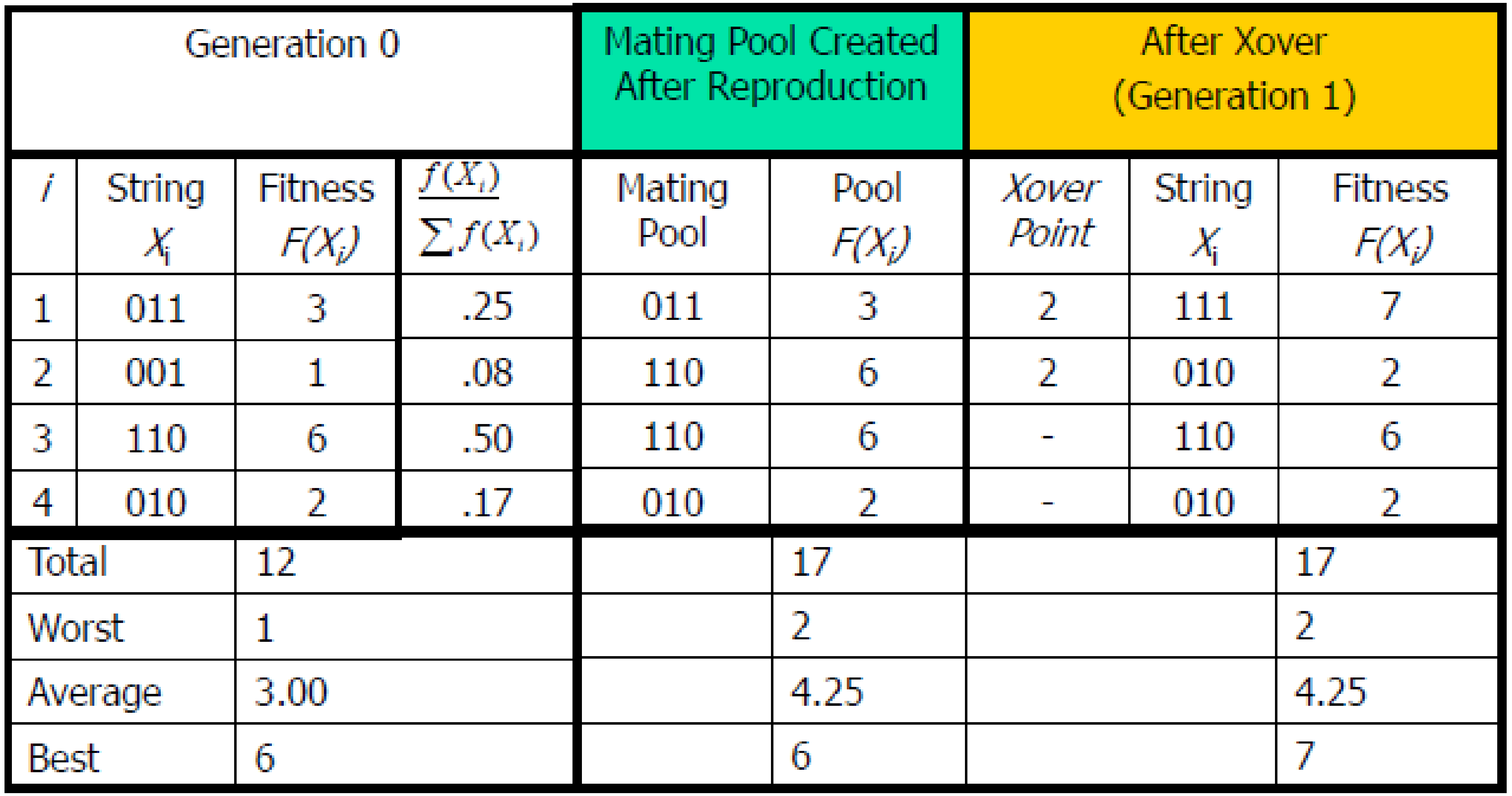
Fixed Point Xover

- Two parents are chosen based on theirs fitness i.e. 011 and 110 are selected.
- Randomly select a point between 0 and L-1 (L = length of the string (3)).
- Suppose the point is 2.



After Xover (Generation 1)		
Xover Point	String X_i	Fitness $F(X_i)$
2	111	7
2	010	2
-	110	6
-	010	2
Total		17
Worst		2
Average		4.25
Best		7

The Result



Generation 0				Mating Pool Created After Reproduction		After Xover (Generation 1)		
i	String X_i	Fitness $F(X_i)$	$\frac{f(X_i)}{\sum f(X_i)}$	Mating Pool	Pool $F(X_i)$	*Xover Point*	String X_i	Fitness $F(X_i)$
1	011	3	.25	011	3	2	111	7
2	001	1	.08	110	6	2	010	2
3	110	6	.50	110	6	-	110	6
4	010	2	.17	010	2	-	010	2
Total		12			17			17
Worst		1			2			2
Average		3.00			4.25			4.25
Best		6			6			7

Example 2: Traveling Salesman Problem (TSP)

- The task: Given a finite number of cities , N and the cost of travel (or distance) between each pair of cities, we need to find the cheapest way(or shortest route) for visiting each city exactly once and returning to the start point.

TSP applications

- Arranging routes for school buses to pick up children in a school district
- Delivering meals to home-bound people
- Planning truck routes to pick up parcel post and many other

Answer these questions?

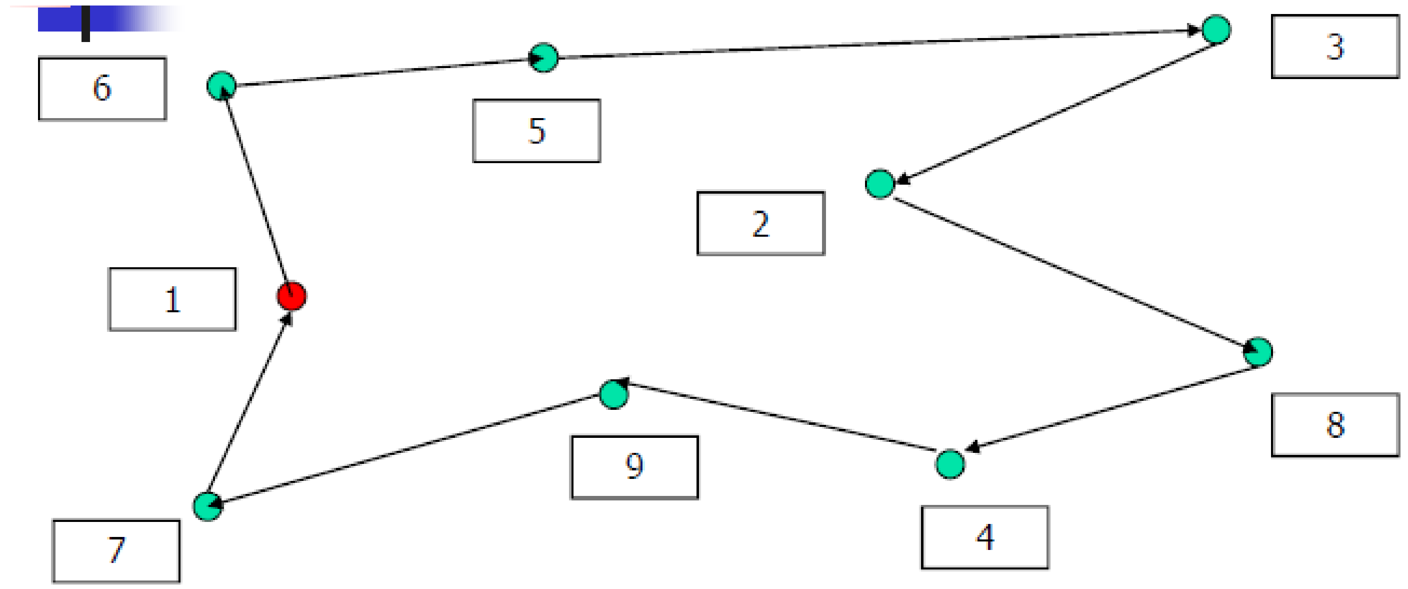
- How do we represent the problem in chromosomes like structure ?
- What should be the initial value of Num. Of Population N, Xover Probability P_c, Mutation Probability P_m and Num. Of generation gen
- What is the fitness function ?
- What kind of stochastic operators we want to use ?
- What will be the termination criterion ?

Solution

- Suppose we have 9 cities, so the chromosomes would be like ..

1	6	5	3	2	8	4	9	7

- the path representation

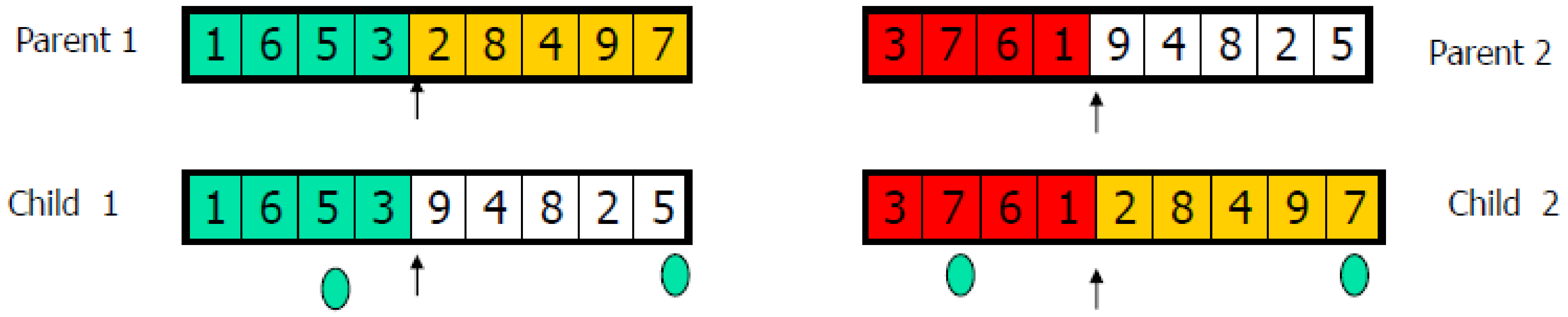


The Fitness Function

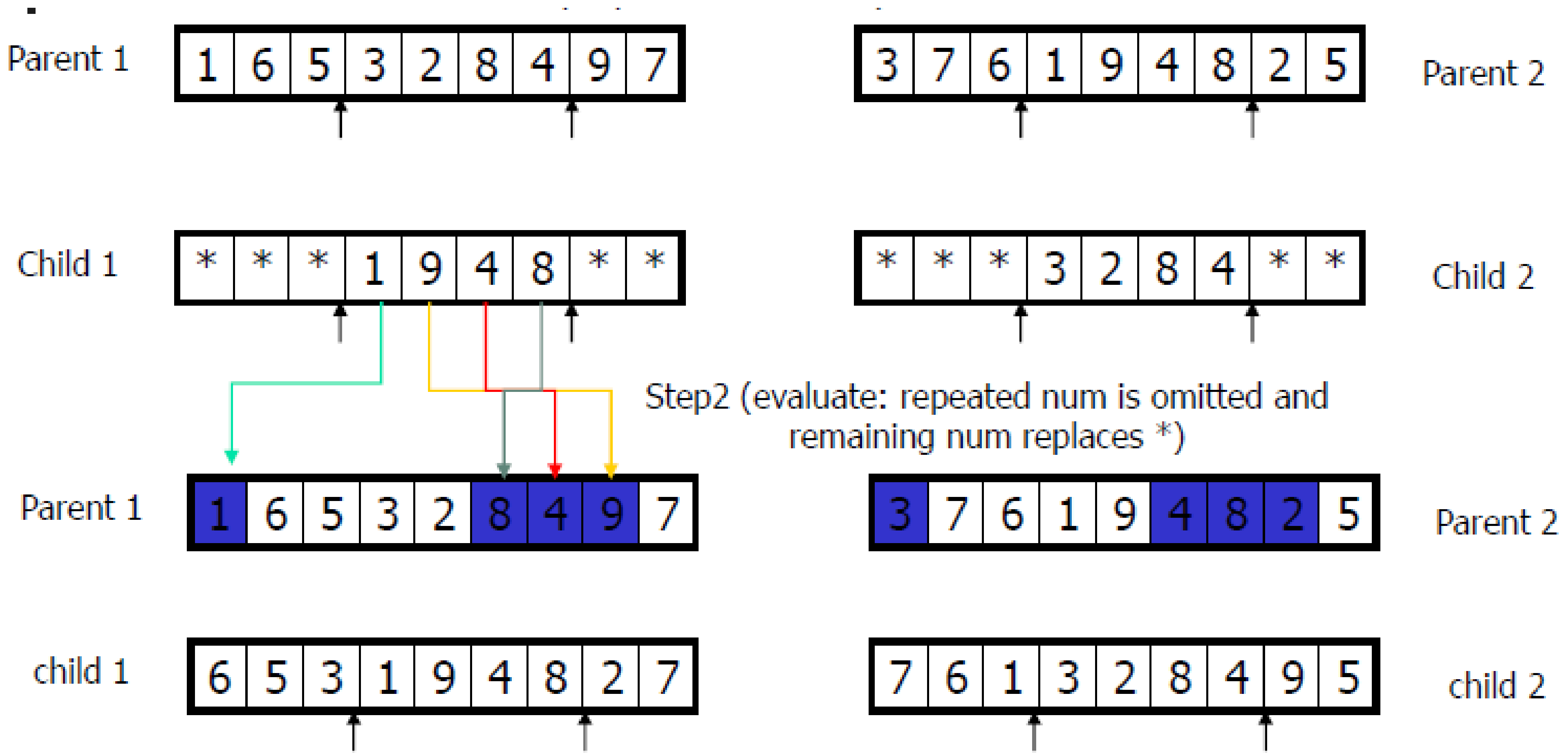
- If we want to find the shortest path, then distance between each pair of cities would be the fitness function the shorter the route the fitter the chromosomes
- If we want to find the optimal cost, then fare/expenses between each pair of cities would be the fitness function.

Stochastic Operators (crossover)

- The crossover in its classical form cannot be directly applied to TSP. A simple exchange of parts between two parents would produce illegal routes containing duplicates and omissions – some cities would be visited twice while some others would not be visited at all as shown below:



CROSSOVER for TSP



Step2 (evaluate: repeated num is omitted and remaining num replaces *)