Introduction To Genetic Algorithms

Cse634

DATA MINING

Professor Anita Wasilewska
Computer Science Department
Stony Brook University

Overview

- Introduction To Genetic Algorithms (GA)
- GA Operators and Parameters
- Genetic Algorithms To Solve The Traveling Salesman Problem (TSP)
- Summary

History Of Genetic Algorithms

- “Evolutionary Computing” was introduced in the 1960s by **I. Rechenberg**

- **John Holland** wrote the first book on **Genetic Algorithms** ‘Adaptation in Natural and Artificial Systems’ in 1975

- In 1992 **John Koza** used **genetic algorithm** to evolve programs to perform certain tasks
- He called his method “Genetic Programming”

What Are Genetic Algorithms?

- What exactly are Genetic Algorithms?
- As the name suggests, Genetic Algorithms borrow their basic working principle from natural genetics

Genetic Algorithms are search and optimization techniques based on **Darwin's Principle** of Natural Selection

Darwin's Principle Of Natural Selection

- IF there are organisms that reproduce, and
- IF offsprings inherit traits from their progenitors, and
- IF there is variability of traits, and
- IF the environment cannot support all members of a growing population,
- **THEN** those members of the population with less-adaptive traits (determined by the environment)

  will die out, and
- **THEN** those members with more-adaptive traits (determined by the environment) will thrive

The result is the evolution of species

Basic Idea of Principle of Natural Selection

“Select The Best,

Discard The Rest”

An Example of Natural Selection

- **Giraffes have long necks**

Giraffes with slightly longer necks could feed on leaves of higher branches when all lower ones had been eaten off

→ **They had a better chance of survival.**

→ **Favorable characteristic propagated through generations of giraffes.**

→ **Now, evolved species has long necks.**

NOTE: **Longer necks** may have been a deviant characteristic (mutation) **initially** but since it was **favorable**, was propagated over generations. Now an established trait.

So, some mutations are beneficial

Evolution Through Natural Selection

Initial Population Of Animals

Struggle For Existence-**Survival Of the Fittest**

Surviving Individuals **Reproduce**, **Propagate** Favorable Characteristics

Millions Of Years

Evolved Species

(Favorable Characteristic Now A Trait Of Species)

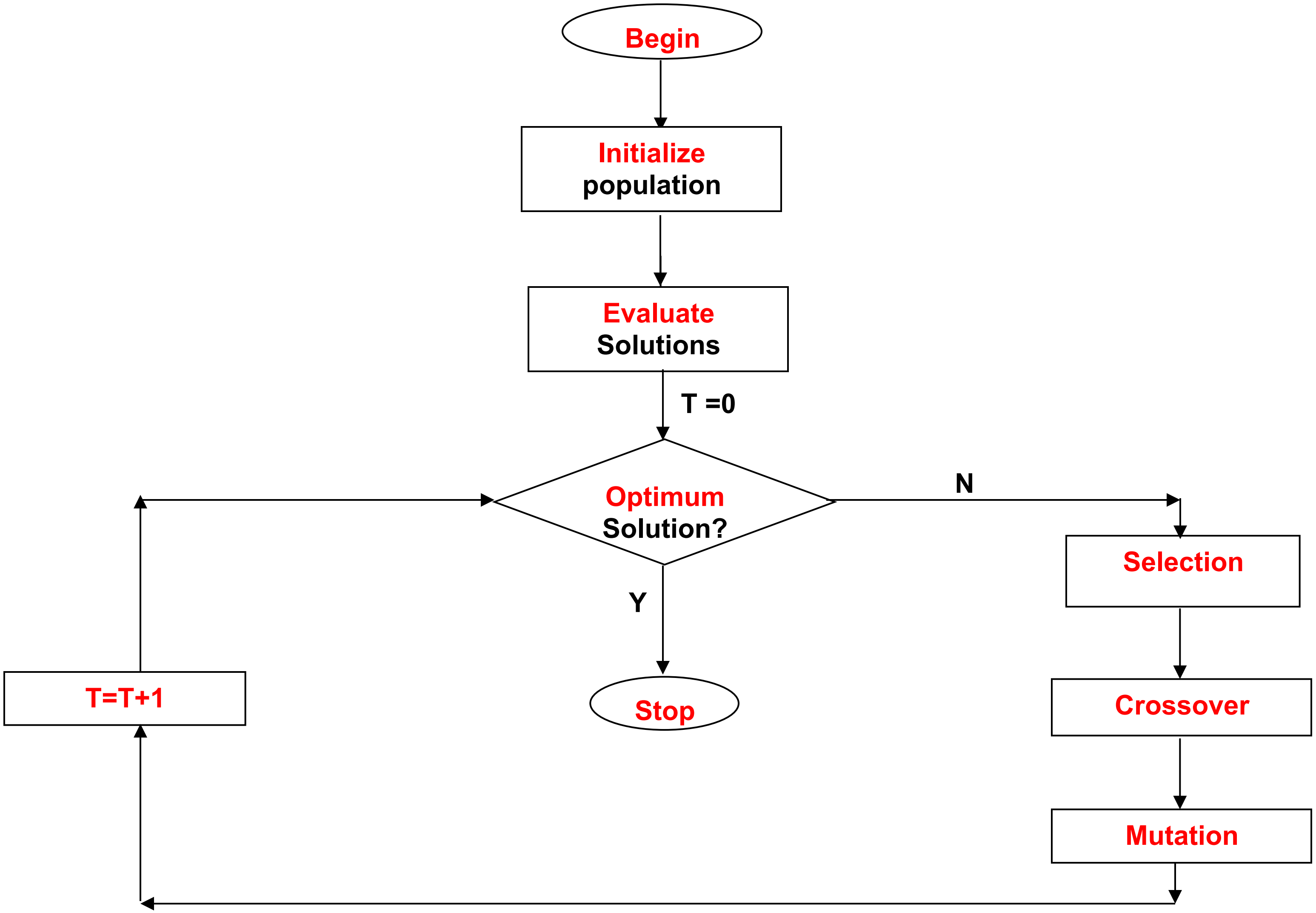
Genetic Algorithms implement

Optimization Strategies

by simulating

evolution of species through natural selection

Working Mechanism Of GA



Simple Genetic Algorithm

```
Simple_Genetic_Algorithm()
    {
    Initialize the Population;
    Calculate Fitness Function;

    While(Fitness Value != Optimal Value)
        {
        Selection;//Natural Selection, Survival Of
   Fittest
        Crossover;//Reproduction, Propagate favorable
   characteristics

        Mutation;//Mutation
        Calculate Fitness Function;
        }
    }
```

Nature to Computer Mapping

Nature	Computer
Population	Set of solutions.
Individual	Solution to a problem.
Fitness	Quality of a solution.
Chromosome	Encoding for a Solution.
Gene	Part of the encoding of a solution.
Reproduction	Crossover

GA Operators and Parameters

Encoding

ENCODING is a process of representing the solution in the form of a **string** that conveys the **necessary information**

- Just as in a chromosome, each gene controls a **particular characteristic** of the individual, similarly, each bit in the **string** represents a **characteristic** of the **solution**

Encoding Methods

- **Binary Encoding** – Most common method of encoding.
- **Chromosomes** are strings of 1s and 0s and each **position** in the **chromosome represents** a particular characteristic of the problem.

Chromosome A	101100101100101011100101
Chromosome B	111111100000110000011111

Encoding Methods

- **Permutation Encoding** – Useful in ordering problems such as the Traveling Salesman Problem (**TSP**)
- In **TSP** **every chromosome is a string of numbers, each of which represents a city to be visited**.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Encoding Methods

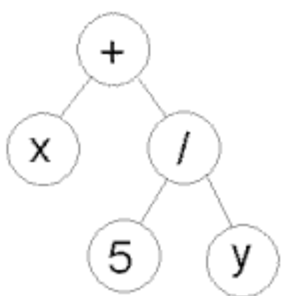
- **Value Encoding** – Used in problems where complicated values, such as **real numbers**, are used and where **binary encoding** would not suffice

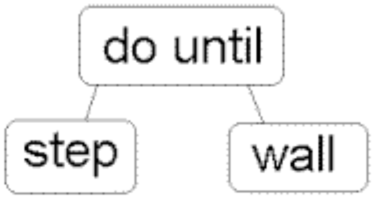
Good for some problems, but often necessary to develop some specific crossover and mutation techniques for these chromosomes.

Chromosome A	1.235 5.323 0.454 2.321 2.454
Chromosome B	(left), (back), (left), (right), (forward)

Encoding Methods

- **Tree Encoding** – this encoding is used mainly for evolving programs or expressions, i.e. for Genetic Programming, or Classification
- **Tree Encoding** - every chromosome is a tree of some objects, such as values/arithmetic operators or commands in a programming language or a **rule**
- (+x(/5y)) (do_until step wall)
-





Genetic Algorithm Operations

Initialization

Selection

Recombination

Reproduction

Termination

Initialization

Individual solutions are randomly generated to form an initial population

- Traditionally, the population is generated randomly,
  - covering the

entire range of possible solutions (the search space)

Fitness Function

A fitness function quantifies the **optimality** of a solution (chromosome) so that that particular solution may be **ranked** against all the other solutions.

- **A fitness value** is assigned to each solution depending on **how close** it actually is to solving the problem

- **Ideal fitness function** **correlates closely to goal** and is quickly computable.

- In **TSP**, **f(x)** is the **sum of distances** between the cities in solution
- The **lesser** the value, the **fitter** the solution is

Recombination

Recombination is a process that **determines** which solutions are to be **preserved** and allowed to **reproduce** and which ones deserve to **die out**

- **The primary objective** of the recombination operator is to emphasize the good solutions and eliminate the bad solutions in a population, while keeping the **population size constant**
- "Selects The Best, Discards The Rest"
- "Recombination" **is different from** "Reproduction"

Recombination

- **Identify** the **good** solutions in a population.

- **Make multiple** copies of the good solutions.

- **Eliminate bad** solutions from the population so that **multiple copies** of good solutions can be **placed** in the population

Selection Methods

There are many different techniques which a genetic algorithm can use to select the individuals to be copied over into the next generation

Listed are some of the most commonly used:

- Roulette-Wheel Selection
- Tournament Selection
- Elitist Selection
- Rank Selection
- Hierarchical Selection

Roulette Wheel Selection

- **Each current string** in the population has a slot assigned to it which is in **proportion** to it's fitness

- **We spin** the weighted roulette wheel thus defined

  n times (where *n* is the total number of solutions)

- Each time Roulette Wheel **stops**, the string corresponding to that slot is created

Strings that are fitter are assigned a larger slot and hence have a better chance of appearing in the new population

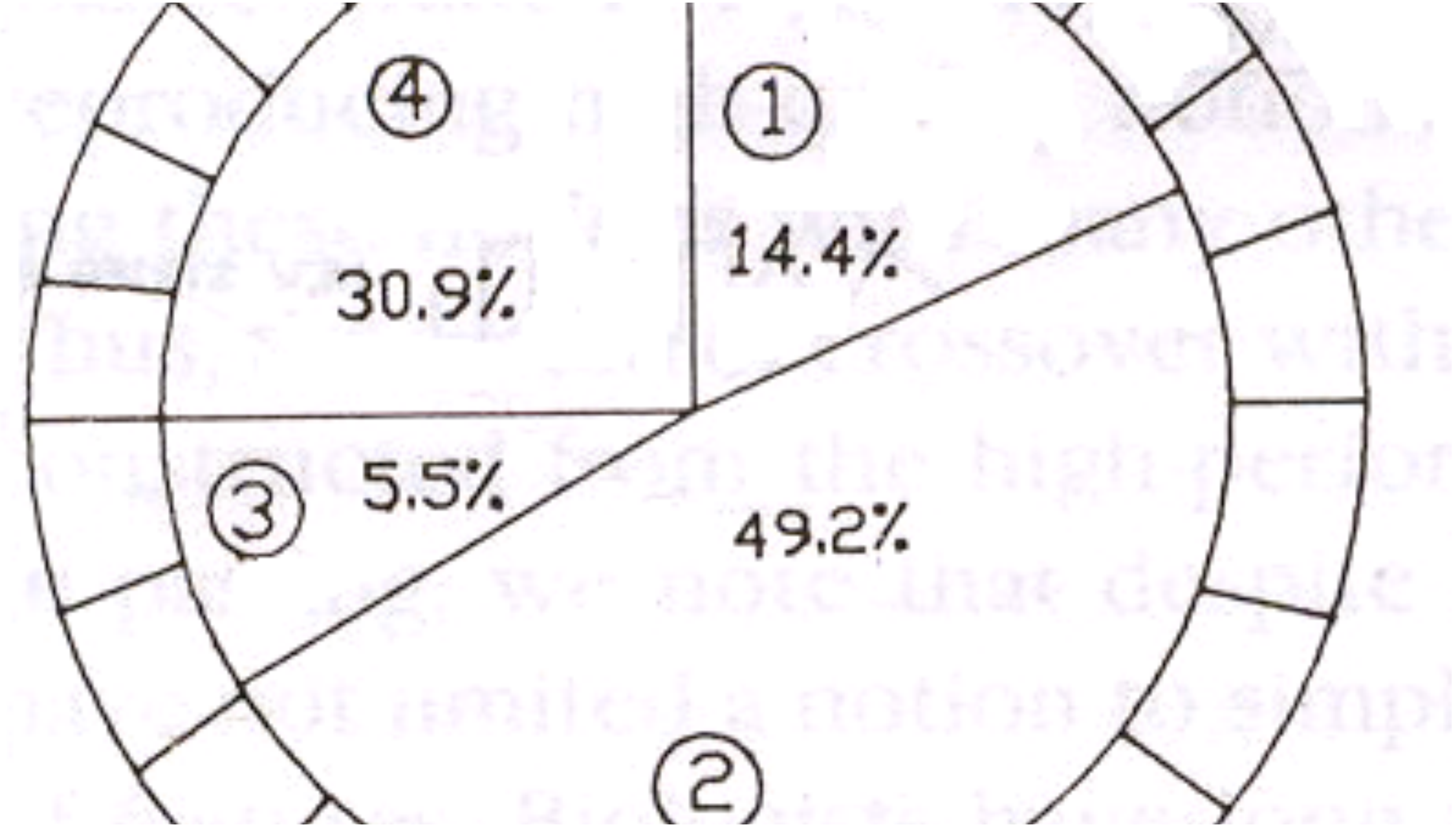
Example Of Roulette Wheel Selection

$\text{Prob}_i = f(i) / \sum_i f(i)$

No.	Initial Population	Fitness f(i)	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total $\sum_i f(i)$		1170	100.0

Roulette Wheel For Example

We spin 4 times



Tournament Selection

GA runs a "tournament" among a few individuals chosen at random from the **population** and selects the winner (the one with the best fitness) for **crossover**

Two chromosomes are picked **out of the pool**, their fitness is compared, and **the better** is permitted to **reproduce**

- Deterministic tournament selection selects the **best individual** in each tournament

- Independent of Fitness function

ADVANTAGE: Decreases computing time, Works on parallel architecture.

Tournament Selection (Pseudo Code)

TS_Procedure_nonDeterministic

{

1. choose k (the tournament size) individuals from the population **at random**

2. choose the **best individual** from pool/tournament with probability p

3. choose the **second best** individual with probability p*(1-p)

4. choose the **third best** individual with probability p*((1-p)^2)

and so on...

}

Reference: wikipedia

Hierarchical Selection

Individuals go through **multiple rounds** of selection each generation

Lower-level **evaluations** are faster and less discriminating

Those that **survive** to higher levels are **evaluated** more rigorously

ADVANTAGE: Efficient usage of computing time by weeding out non-promising candidate chromosomes

Rank Selection

Rank selection first ranks the population and then every chromosome receives fitness from this ranking

Selection is based on this ranking rather than **absolute differences** in fitness

The worst will have fitness 1, **second worst** will have fitness 2, etc…

and **the best** will have fitness N

(where N is the number of chromosomes in population)

ADVANTAGE: Preserves genetic diversity (by preventing dominance of "fitter" chromosomes)

Reproduction

GA Reproduction operators:

Crossover

Mutation

Elitism

Crossover

Crossover is a GA **operator** in the **process** in which **two chromosomes** (strings) **combine** their genetic material (bits) to **produce** a new offspring which possesses both their characteristics

- The two **chromosomes** are picked from the **mating pool** to **crossover** to **produce** a new offspring

We also introduce a **crossover probability** to indicate a ratio of how many **chromosome** will be picked for **crossover**

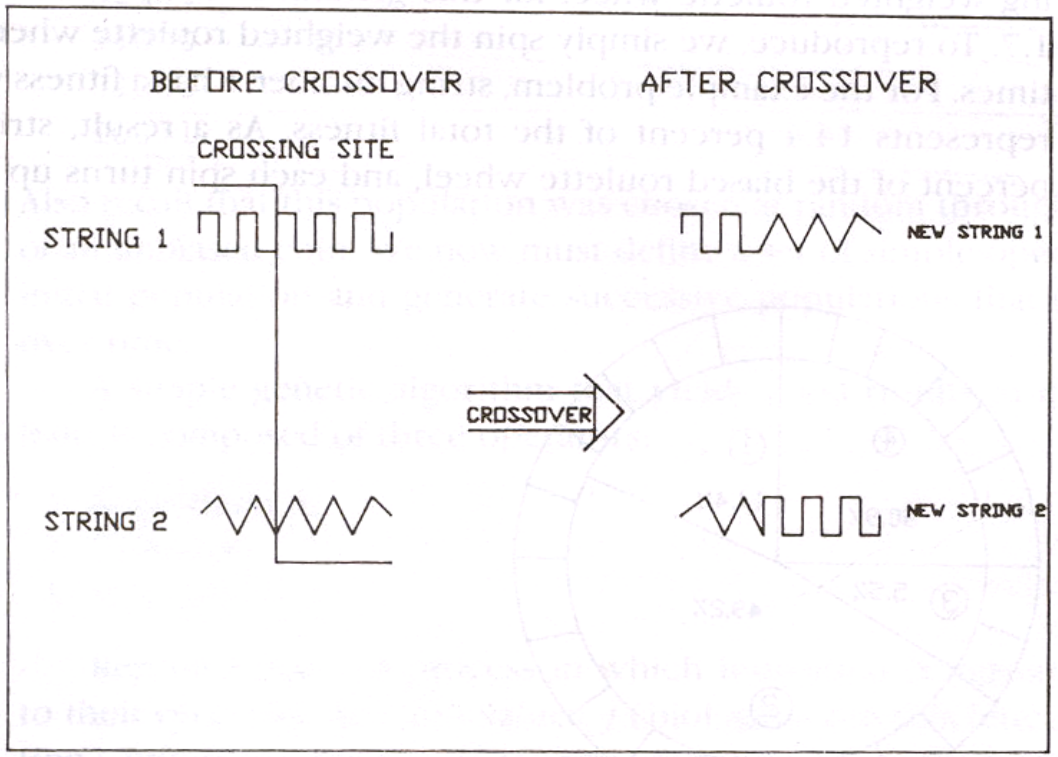
They are usually picked by following **selection** criteria

The method chosen for **crossover** depends on a chosen

crossover encoding method

Crossover Methods

- **Single Point Crossover-** a random point is chosen on the individual chromosomes (strings) and the genetic material is exchanged at this point



Crossover Methods

- **Single Point Crossover**

Chromosome1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

Crossover Methods

- **Two-Point Crossover-** two random points are chosen on the individual chromosomes (strings) and the genetic material is **exchanged** at these points

Chromosome1	**11011 | 00100 | 110110**
Chromosome 2	**10101 | 11000 | 011110**
Offspring 1	**10101 | 00100 | 011110**
Offspring 2	**11011 | 11000 | 110110**

NOTE: These chromosomes are different from the last example

Crossover Methods

- **Uniform Crossover-** each gene (bit) is selected randomly from one of the corresponding genes of the parent chromosomes

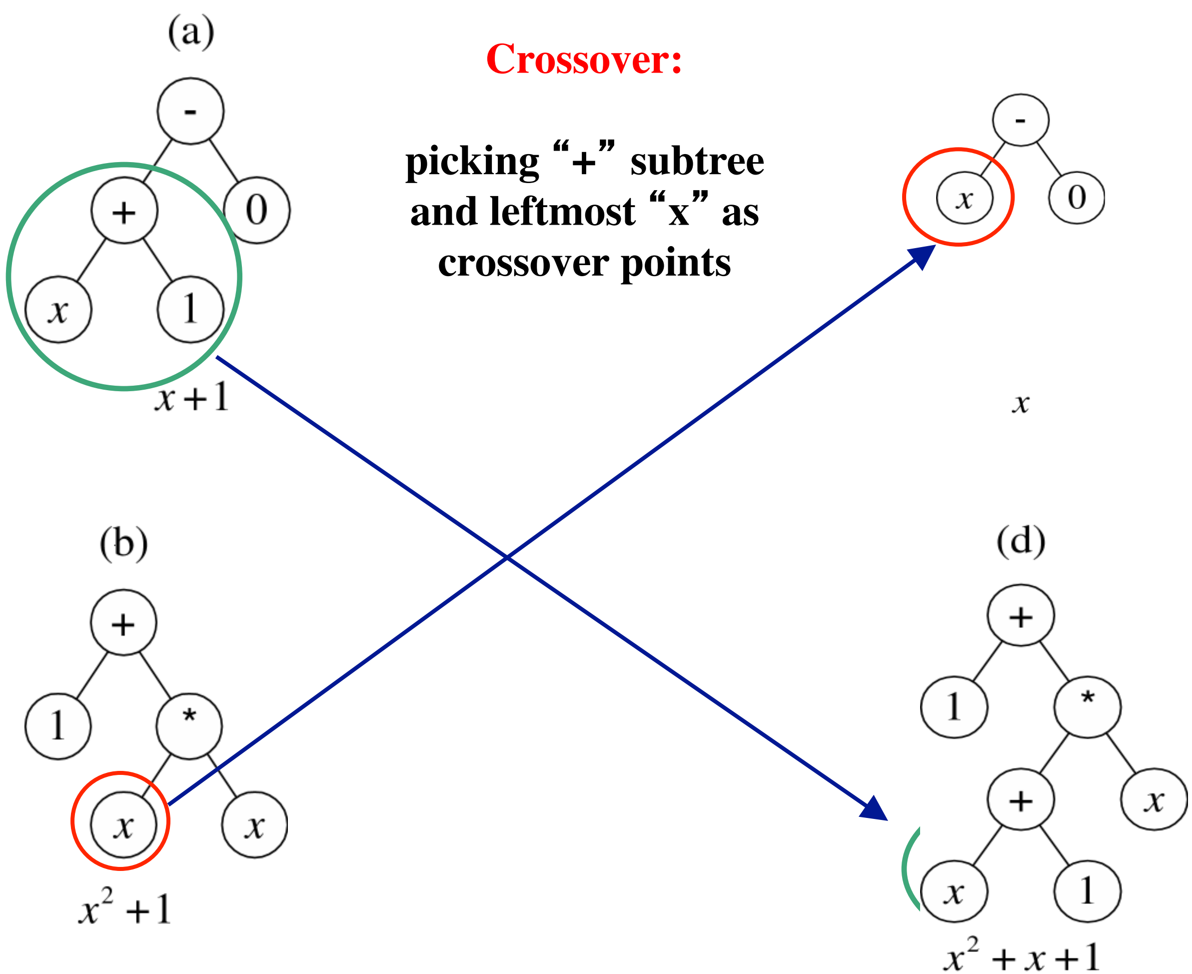
Chromosome1	**11011 | 00100 | 110110**
Chromosome 2	**10101 | 11000 | 011110**
Offspring	**10111 | 00000 | 110110**

NOTE: Uniform Crossover yields ONLY 1 offspring

Trees Crossover

Crossover:

picking "+" subtree and leftmost "x" as crossover points



Crossover

- **Crossover** between **2 good solutions**
- **MAY NOT ALWAYS** yield a better or as good a solution

- Since parents **are good**, probability of the child being **good** is high

- **If offspring** is not good (poor solution), it will be removed in the next iteration during **Selection**

Elitism

Elitism is a method in which copies the best chromosome are **added** to the new offspring population **before** crossover and mutation

- When creating a **new population** by crossover or mutation the **best chromosome** might be lost
- **Elitism** lets **GA** to **retain** some number of the **best** individuals at each generation
- It has been found that **elitism** significantly **improves** performance

Mutation

Mutation is the process by which a string is deliberately changed so as to maintain diversity in the **population set**

We saw in the giraffes' example, that mutations could be beneficial

Mutation Probability (or Ratio) is a measure of the likeness that random elements of a chromosome will be changed

Example

Let a chromosome be encoded as a binary string of length 100

1% mutation probability means that **1** out of **100** bits (on average) picked at random will be changed

Mutation

A common method of **implementing** the **mutation operator** involves generating a random variable for **each bit** in a sequence

This random variable tells a particular bit will be modified

Example of Mutation

- For chromosomes using Binary Encoding, **randomly selected** bits are **inverted**

Offspring	**11011 00100 110110**
Mutated Offspring	**11010 00100 100110**

NOTE: The number of bits to be inverted depends on the Mutation Probability

Crossover vs Mutation

Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem

Exploitation: Optimising within a promising area, i.e. using information

There is co-operation AND competition between them

Crossover is explorative, it makes a **big jump** to an area somewhere **"in between"** two (parent) areas

Mutation is exploitative, it creates **random small diversions**, thereby staying near (in the area of) the parent

Simple Genetic Algorithm (Reproduction Cycle)

Select parents for the mating pool

(size of mating pool = population size)

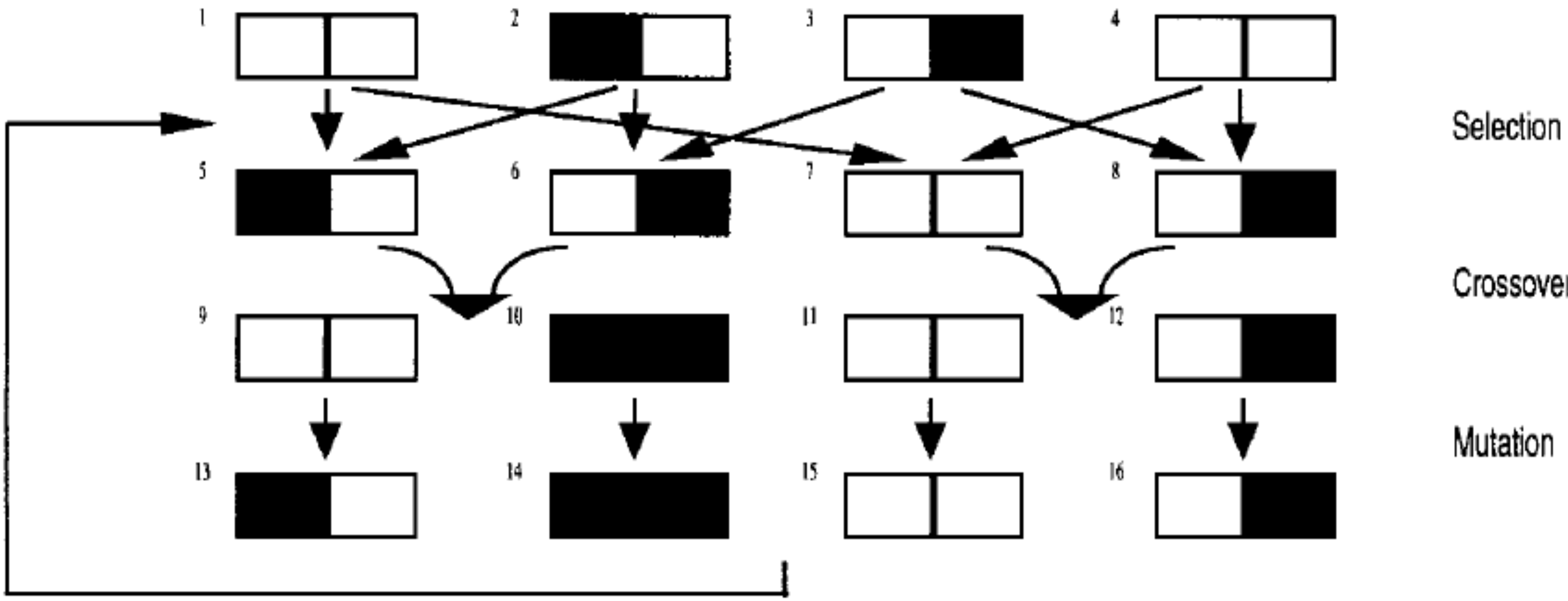
Shuffle the mating pool

For each consecutive pair apply crossover with probability P_c, otherwise copy parents

For each offspring apply mutation (bit-flip with probability P_m independently for each bit)

Replace the whole population with the resulting offsprings

One generation of a genetic algorithm, consisting of - from top to bottom - selection, crossover, and mutation stages



Genetic Algorithm Short

- **INITIALIZATION**

- At the beginning of a run of a **Genetic Algorithm** an
- INITIAL POPULATION of *random* chromosomes is created

- The INITIAL POPULATION depends on the nature of the problem, but typically contains several hundreds or thousands of possible chromosomes (possible solutions)

- Often the INITIAL POPULATION covers the entire range of possible solutions (the *search space*)

- Sometimes the solutions (chromosomes) may be "seeded" in areas where optimal solutions are likely to be found

Basic Genetic Algorithm

Assume that there are *N* chromosomes in the INITIAL POPULATION

REPRODUCTION CYCLE

1. **Test** each chromosome to see how good it is, i.e assign a *fitness score* accordingly
2. **Select two members** from the current population

The **chance** of being selected is proportional to the chromosomes fitness

Roulette wheel **selection** is a commonly used method

Basic Genetic Algorithm

3. **Perform** CROSSOVER on the **selected** pair

4. **Apply** MUTATION to each **offspring**

5. **Repeat** steps **2, 3, 4** **until** a new population of N chromosomes has been created, i.e. we keep the population size **constant**

The REPRODUCTION CYCLE is **repeated** **until** a TERMINATION CONDITIONS has been reached

Termination Conditions

- Common TERMINATION CONDITIONS are:
- A solution is **found** that satisfies problem **criteria**
- Fixed number of generations is **reached**
- Allocated budget (computation time/money) is **reached**
- The highest ranking solution's fitness has **reached** a plateau
- Manual inspection
- **Combinations** of the above

GA Short Overview

- SELECTION
- During each successive generation, a portion of the existing population is **selected** through a *fitness-based* process measured by a **fitness function**
- The fitness function is always problem dependent
- For each new chromosome (solution) to be produced, a pair of "parent" chromosomes is **selected** from the pool selected previously

Genetic Algorithm Short Overview

- REPRODUCTION CYCLE
- The new chromosome (solution) is **produced** by applying operations of **crossover** and **mutation**
- New parents are selected for each new child, and
- **the process** continues **until** a new population of chromosomes (solutions) of appropriate constant size is **generated**
- It is possible to use **other operators** such as regrouping, colonization-extinction, or migration

Parameters

- Parameters such as the crossover probability, mutation probability and population size are used (and tuned) to find reasonable settings for the problem

- A very small mutation rate may lead to genetic drift
- A recombination rate that is too high may lead to **premature convergence** of the genetic algorithm

- A mutation rate that is too high may lead to loss of **good** solutions, unless we employ the elitist selection

Genetic Algorithms To Solve The Traveling Salesman Problem (TSP)

Traveling Salesman Problem

We are given a set of cities and a symmetric distance matrix that indicates the **cost** of travel from each city to every other city

The goal is to **find the shortest circular tour**, visiting every city **exactly once**, so as to **minimize the total travel cost**, which includes the cost of traveling from the last city back to the first city

Encoding

- **We represent every city with an integer**

- **Consider** 6 Indian cities –

  Mumbai, Nagpur , Calcutta, Delhi , Bangalore and Chennai and assign a number to each.

Mumbai	1	→
Nagpur	2	→
Calcutta	3	→
Delhi	4	→
Bangalore	5	→
Chennai	6	→

Encoding

- Thus a **path** would be represented as a sequence of integers from 1 to 6

- **The path [1 2 3 4 5 6]** represents a path from Mumbai to Nagpur, Nagpur to Calcutta, Calcutta to Delhi, Delhi to Bangalore, Bangalore to Chennai, and finally from Chennai to Mumbai

- This is an example of **Permutation Encoding** as the **position of the elements determines the fitness** of the solution.

Fitness Function

- **The fitness function** is be the **total cost of the tour** represented by each chromosome.

The fitness function is calculated as the **sum of the distances** traversed in each travel segment

The Lesser The Sum, The Fitter The Solution Represented By That Chromosome

Distance/Cost Matrix For TSP

	1	2	3	4	5	6
1	0	863	1987	1407	998	1369
2	863	0	1124	1012	1049	1083
3	1987	1124	0	1461	1881	1676
4	1407	1012	1461	0	2061	2095
5	998	1049	1881	2061	0	331
6	1369	1083	1676	2095	331	0

Cost matrix for six city example.
Distances in Kilometers

Fitness Function

- So, for a chromosome **[4 1 3 2 5 6]**, the total cost of travel or fitness will be calculated as shown below

- **Fitness** *= 1407 + 1987 + 1124 + 1049 + 331+ 2095*

  = 7993 km

- Since our **objective** is to **Minimize the distance**, the lesser the total distance, the **fitter the solution**.

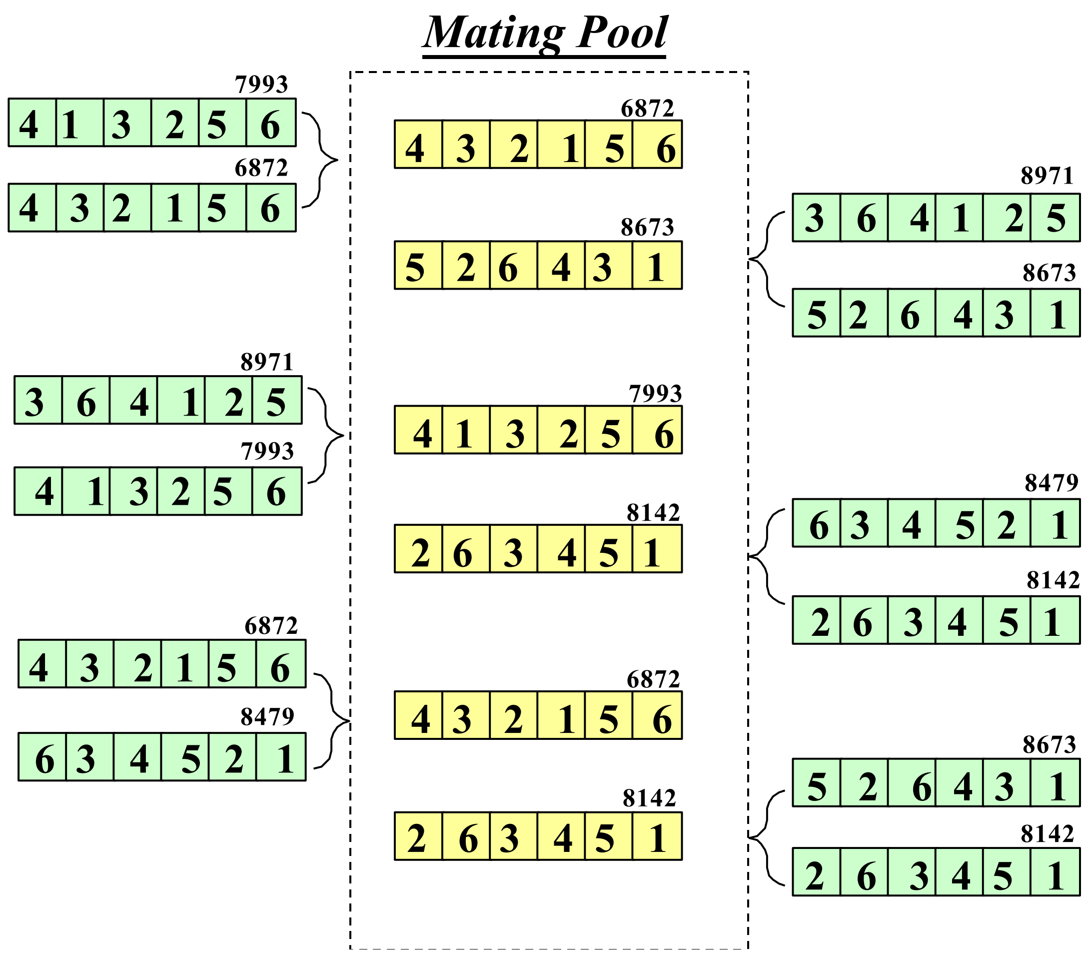
Selection Operator

We use Tournament Selection

As the name suggests **tournaments** are played between **two solutions** and the better solution is chosen and placed in the mating pool

Two other solutions are picked again and another slot in the mating pool is filled up with the **better solution**

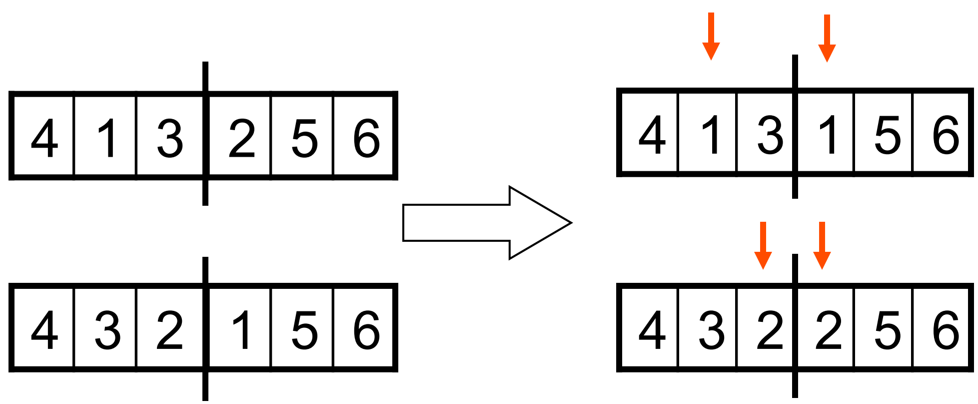
Tournament



Mating Pool
7993
4 1 3 2 5 6
6872
4 3 2 1 5 6
8971
3 6 4 1 2 5
7993
4 1 3 2 5 6
6872
4 3 2 1 5 6
8479
6 3 4 5 2 1
6872
4 3 2 1 5 6
8673
5 2 6 4 3 1
7993
4 1 3 2 5 6
8142
2 6 3 4 5 1
6872
4 3 2 1 5 6
8142
2 6 3 4 5 1
8971
3 6 4 1 2 5
8673
5 2 6 4 3 1
8479
6 3 4 5 2 1
8142
2 6 3 4 5 1
8673
5 2 6 4 3 1
8142
2 6 3 4 5 1

Why we cannot use single-point crossover:

- **Single point crossover** method randomly selects a crossover point in the string and swaps the substrings.
- This may produce some **invalid offsprings** as shown below.



Crossover Operator

- We use the **Enhanced Edge Recombination** operator (T.Starkweather, et al, *'A Comparison of Genetic Sequencing Operators,* International Conference of GAs, 1991*)*

- This operator is different from other genetic sequencing operators in that it emphasizes **adjacency information** instead of the order or position of items in the sequence.

- The algorithm for the **Edge-Recombination Operator** involves constructing an Edge Table first

Edge Table

The **Edge Table** is an *adjacency table* that lists links ***into*** and ***out** of* a city found in the **two parent sequences.**

If an item **is already** in the edge table and we are trying to insert it again, that element of a sequence must be a common edge and is **represented** by inverting it's sign.

Finding The Edge Table

Parent 1

4	1	3	2	5	6

Parent 2

4	3	2	1	5	6

1	4	3	2	5
2	-3	5	1	
3	1	-2	4	
4	-6	1	3	
5	1	2	-6	
6	-5	-4		

Enhanced Edge Recombination Algorithm

1. Choose the initial city from one of the two parent tours. (It can be chosen randomly as according to criteria outlined in ***step 4***). This is the current city

2. Remove all occurrences of the current city from the left hand side of the edge table.(These can be found by referring to the edge-list for the current city)

3. If the current city has entries in it's edge-list, go to ***step 4*** otherwise **go to *step 5***

4. Determine which of the cities in the edge-list of the current city has the fewest entries in it's own edge-list. The city with fewest entries becomes the current city. In case a negative integer is present, it is given preference. Ties are broken randomly. **Go to *step 2*.**

5. If there are no remaining *unvisited* cities*,* then *stop.*
6. Otherwise, randomly choose an *unvisited* city and go to ***step 2.***

Example Of Enhanced Edge Recombination Operator

Step 1

1	4	3	2	5
2	-3	5	1	
3	1	-2	4	
4	-6	1	3	
5	3	2	-6	
6	-5	-4		

4					

Step 2

1	3	2	5
2	-3	5	1
3	1	-2	
4	-6	1	3
5	3	2	-6
6	-5		

4	6				

Example Of Enhanced Edge Recombination Operator

Step 3

1	3	2	5
2	-3	5	1
3	1	-2	
4	1	3	
5	3	2	
6	-5		

4	6	5			

Step 4

1	3	2
2	-3	1
3	1	-2
4	1	3
5	3	2
6		

4	6	5	1		

Example Of Enhanced Edge Recombination Operator

Step 5

1	3	2
2	-3	
3	-2	
4	3	
5	3	2
6		

4	6	5	1	3	

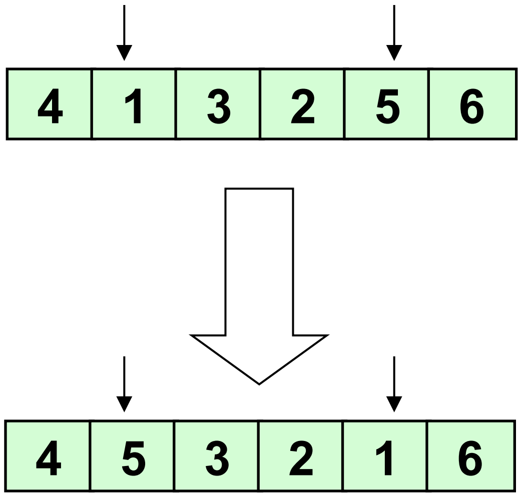
Step 6

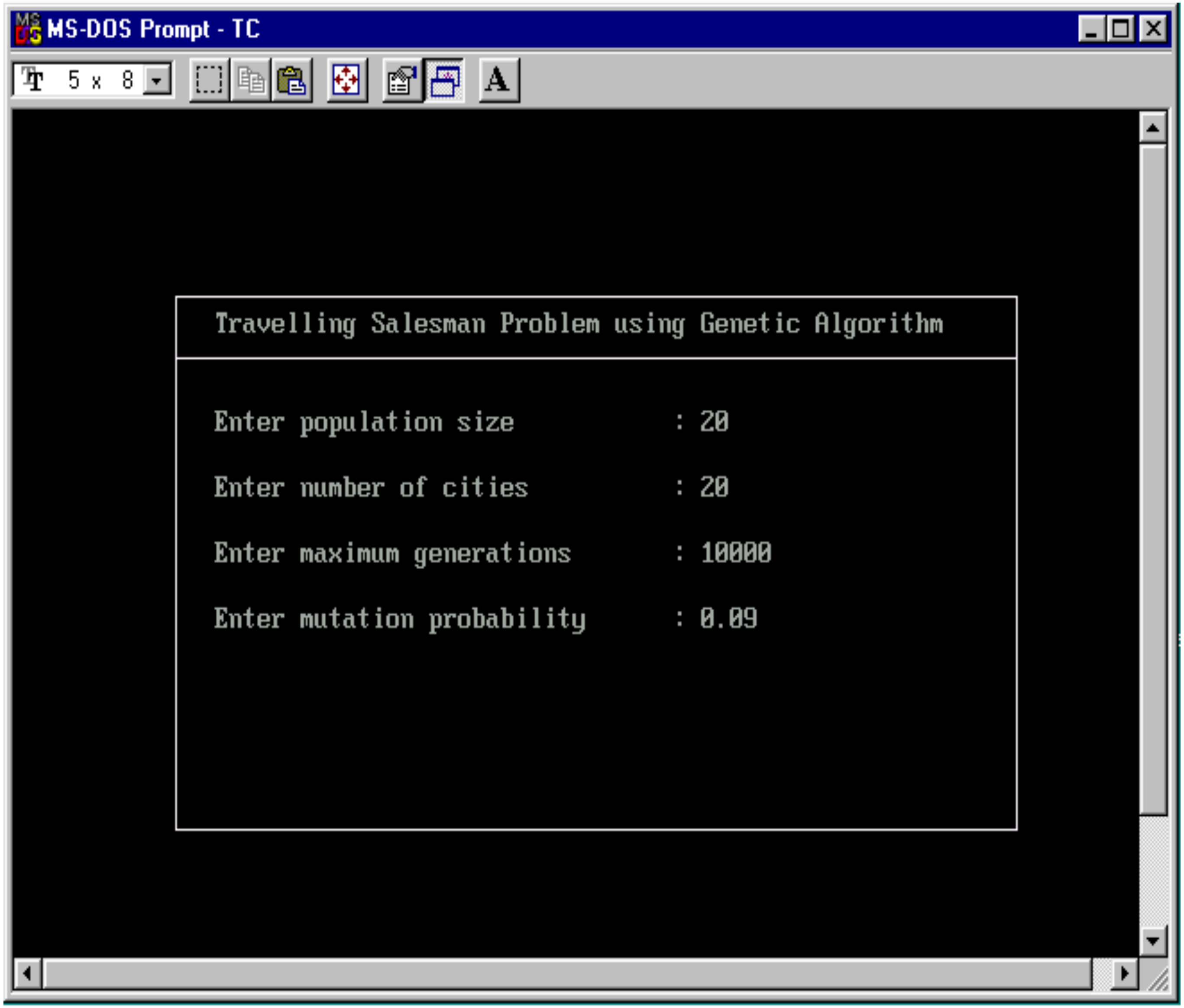
1	2
2	
3	-2
4	
5	2
6	

4	6	5	1	3	2

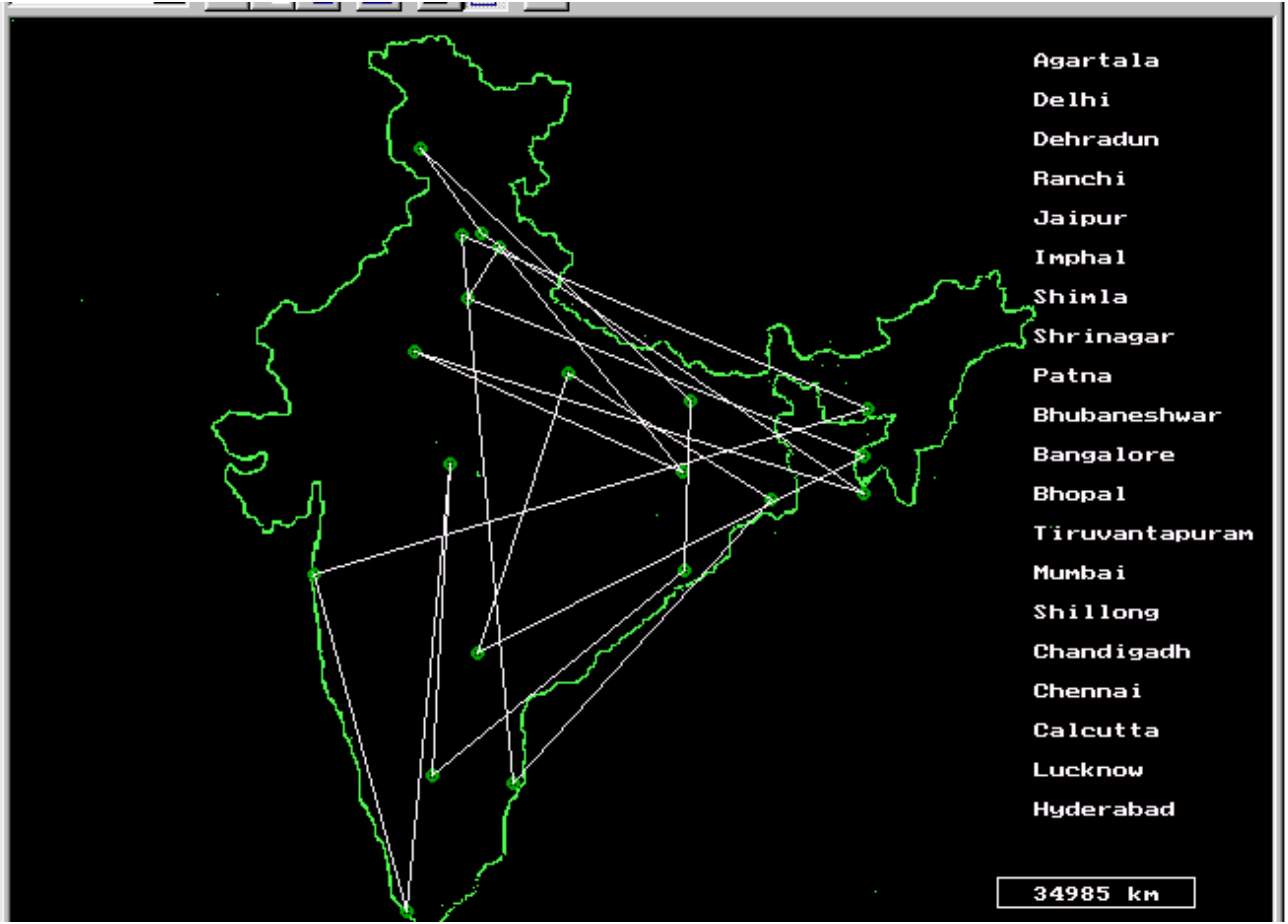
Mutation Operator

- **The mutation operator** induces a change in the solution, so as to maintain diversity in the population and prevent **Premature Convergence**
- We **mutate the string** by randomly selecting any two cities and interchanging their positions in the solution, thus giving rise to a new tour.

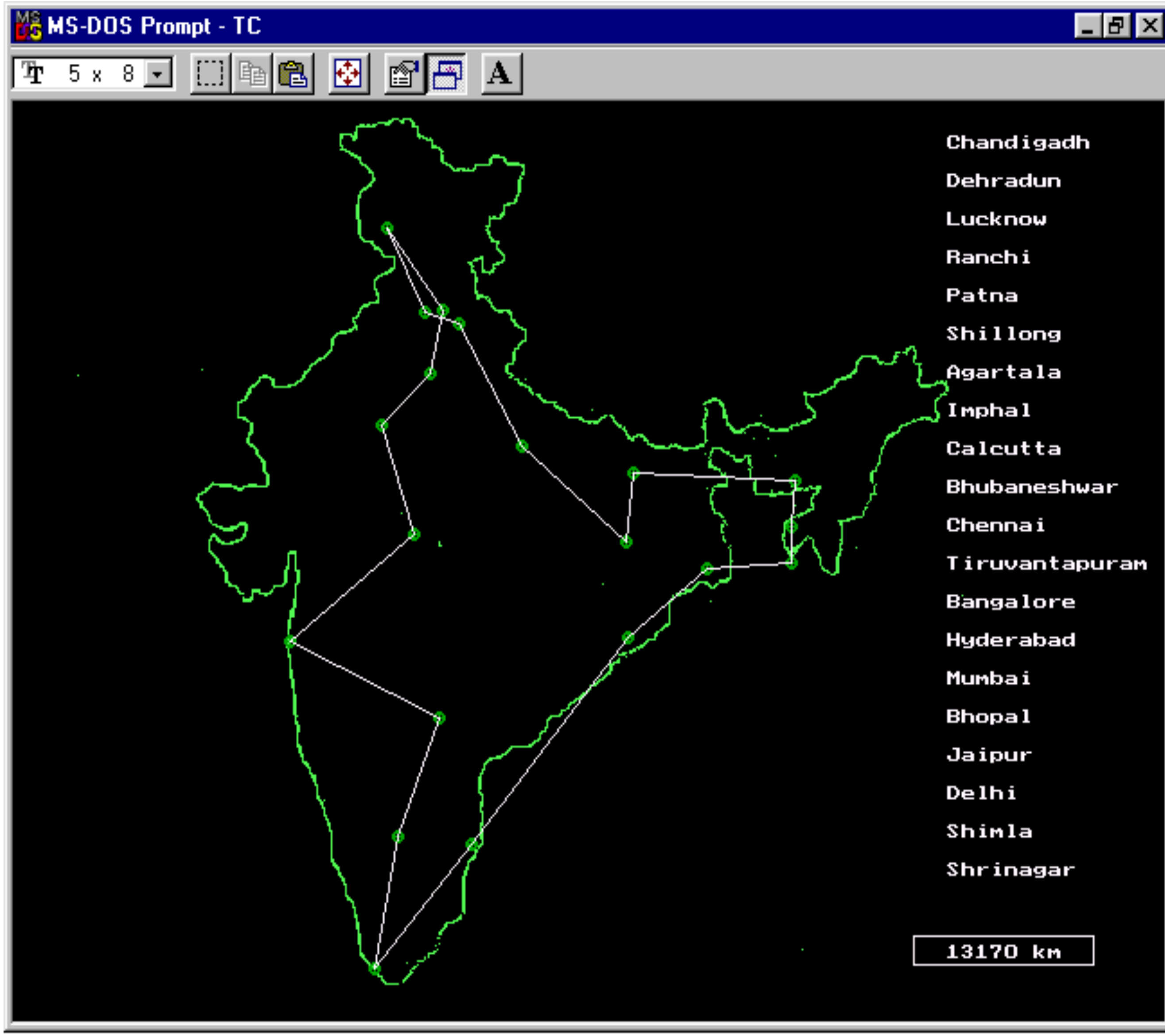




Input To Program



Initial Output For 20 cities : Distance=34985 km
Initial Population



Final Output For 20 cities : Distance=13170 km
Generation 4786

Advantages Of GA

GAs search for the function **optimum** starting from a **population of points** of the function domain, **not a single one**

This characteristic suggests that **GAs** are **global search methods**

- They can, in fact, climb many peaks in parallel, reducing the probability of finding local minima, which is one of the **drawbacks** of traditional **optimization methods**

Advantages of GA

GAs only use the **information** about **the objective function**

GAs do not require knowledge of any other auxiliary information

They allow a **number of problems** to be solved without the need to formulate restrictive assumptions

For this reason, GAs are often called **blind search methods**

Advantages of GA

- **GAs can be easily implemented on parallel machines**

- Since in real-world design **optimization problems**, most computational time is spent in evaluating a solution, with multiple processors all **solutions in a population** can be evaluated in a distributed manner

This reduces substantially the overall **computational time**

Some References

WEBSITES

- www.iitk.ac.in/kangal
- www.math.princeton.edu
- www.genetic-programming.com
- www.garage.cse.msu.edu
- www.aic.nre.navy.mie/galist