Genetic Algorithms Examples + Operators

Sabah Sayed

Department of Computer Science

Faculty of Computers and Artificial Intelligence

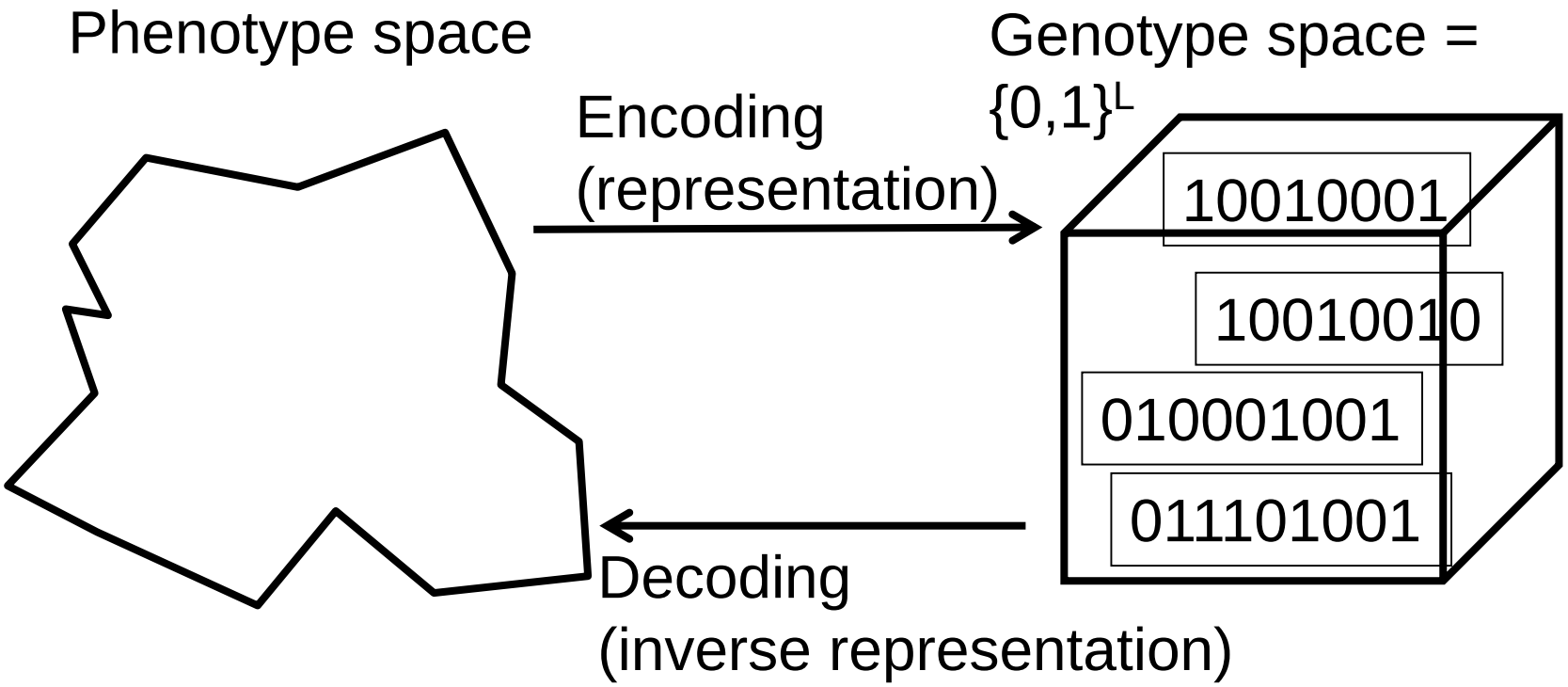
Cairo University

Egypt

Simple GA Representation

Binary Code

Gray Code?



Gray code representation

- Gray coding of integers (still binary chromosomes)
  - "Smoother" genotype-phenotype mapping makes life easier for the GA
  - Gray coding is a mapping that means that *small changes in the genotype cause small changes in the phenotype* (unlike binary coding).
  - It gives stability in exploring the search space.
  - Examples?

Binary	Gray	Binary	Gray
0000	0000	1000	1100
0001	0001	1001	1101
0010	0011	1010	1111
0011	0010	1011	1110
0100	0110	1100	1010
0101	0111	1101	1011
0110	0101	1110	1001
0111	0100	1111	1000

7 0111
8 1000
binary

0100 7
1100 8
Gray

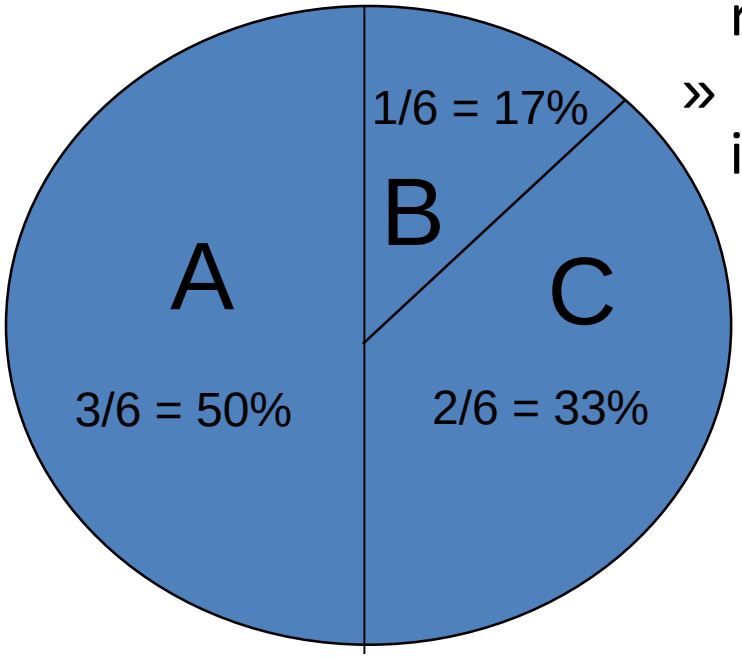
GA Population Initialization

Start with a population of **randomly** generated individuals, or use

- A previously saved population
- A set of solutions provided by a human expert
- A set of solutions provided by another heuristic algorithm

Roulette Wheel Selection

- Main idea: better individuals get higher chance
  - Chances **proportional** to fitness
  - Implementation: Roulette Wheel technique
    - » Assign to each individual a part of the roulette wheel
    - » Spin the wheel n times to select n individuals



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

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.

Roulette Wheel Selection

- Calculate S = the sum of all fitness values.

- Generate a random number R between 0 and S-1.
- Starting from the top of the population, keep adding the fitnesses to the partial sum P, as long as P<=R.

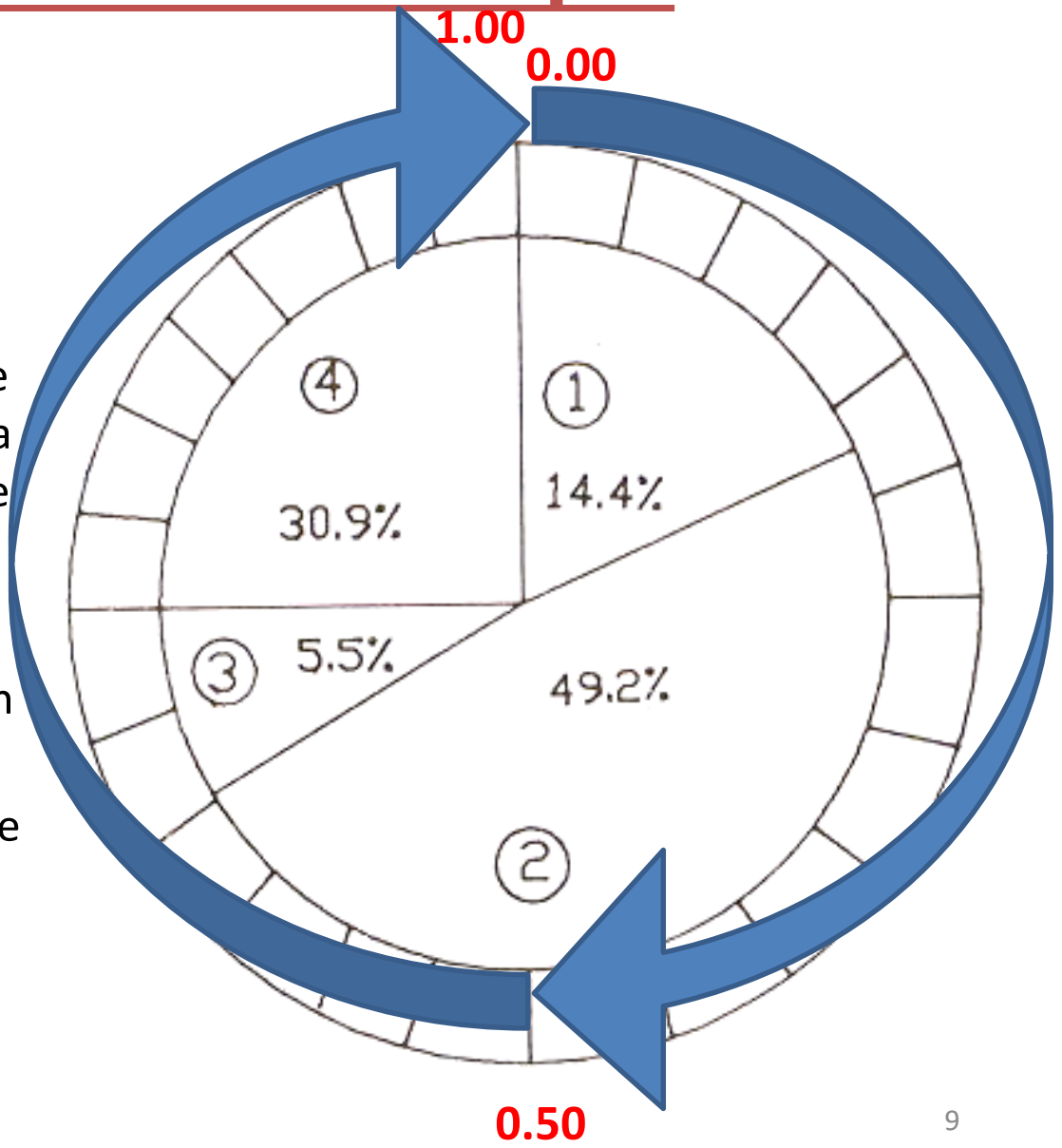
- The individual for which P exceeds R is the chosen individual.

Example Of Roulette Wheel Selection

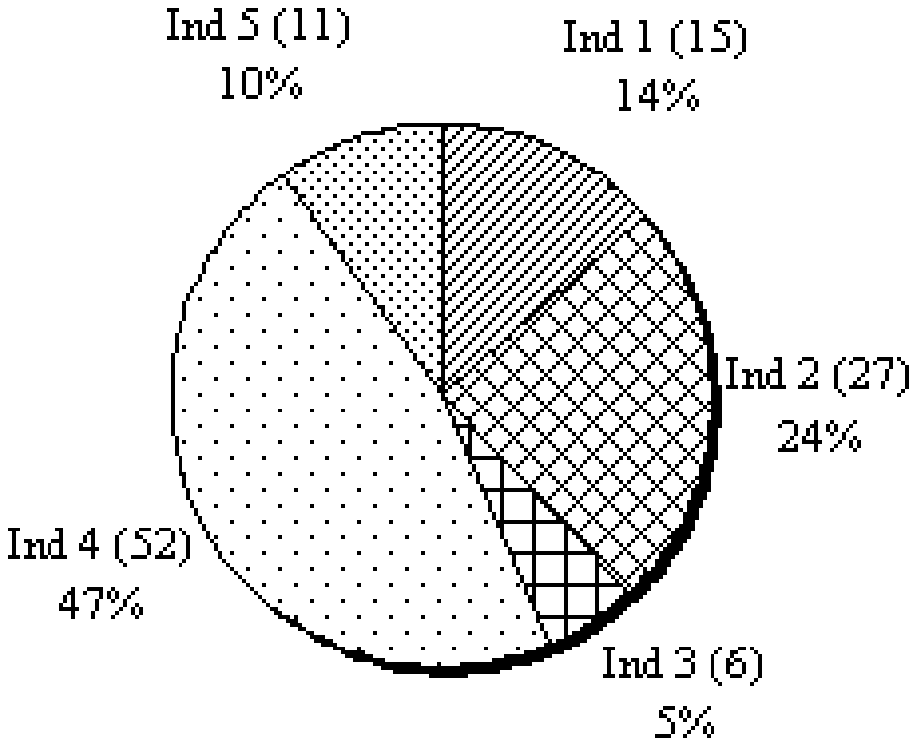
No.	String	Fitness	% Of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

Roulette Wheel Example

- Spinning the wheel:
  - The bigger the area of one player, the higher the chance/probability that the ball will stop in his/her area … so, the higher the chance he/she wins.
- In GA:
  - A random number between 0.00 and 1.00 is generated.
  - The individual/chromosome whose range covers this random number will be selected for further operations.



Population	*Fitness*
Individual 1	15
Individual 2	27
Individual 3	6
Individual 4	52
Individual 5	11



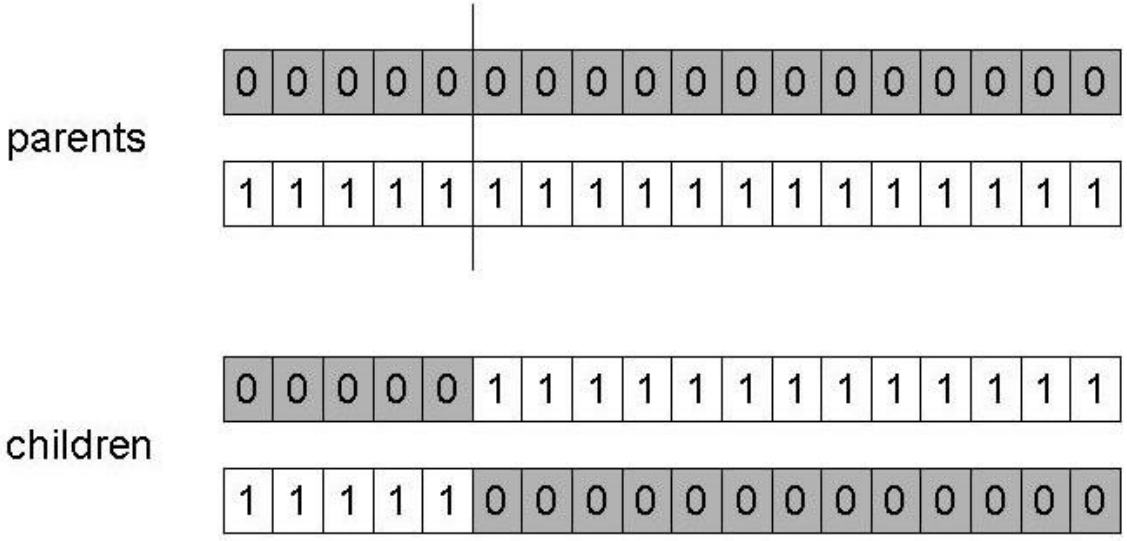
Randomly generated number = 21

Individual 2 is selected

Roulette Wheel Selection

Single point Crossover Operator

- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails

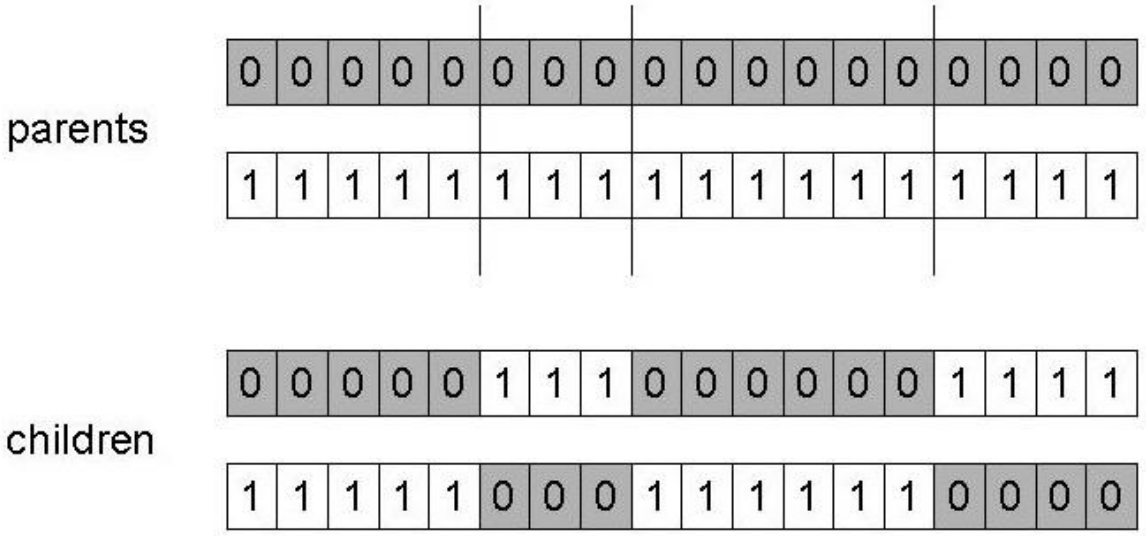


Single point Crossover Operator

- Performance with 1 Point Crossover depends on the order that variables occur in the representation
  - more likely to keep together genes that are near each other
  - Can never keep together genes from opposite ends of string
  - This is known as *Positional Bias*
  - Can be exploited if we know about the structure of our problem, but this is not usually the case

N-point Crossover Operator

- Choose n random crossover points
- Split along those points
- Glue parts, alternating between parents
- Generalisation of 1 point (still some positional bias)



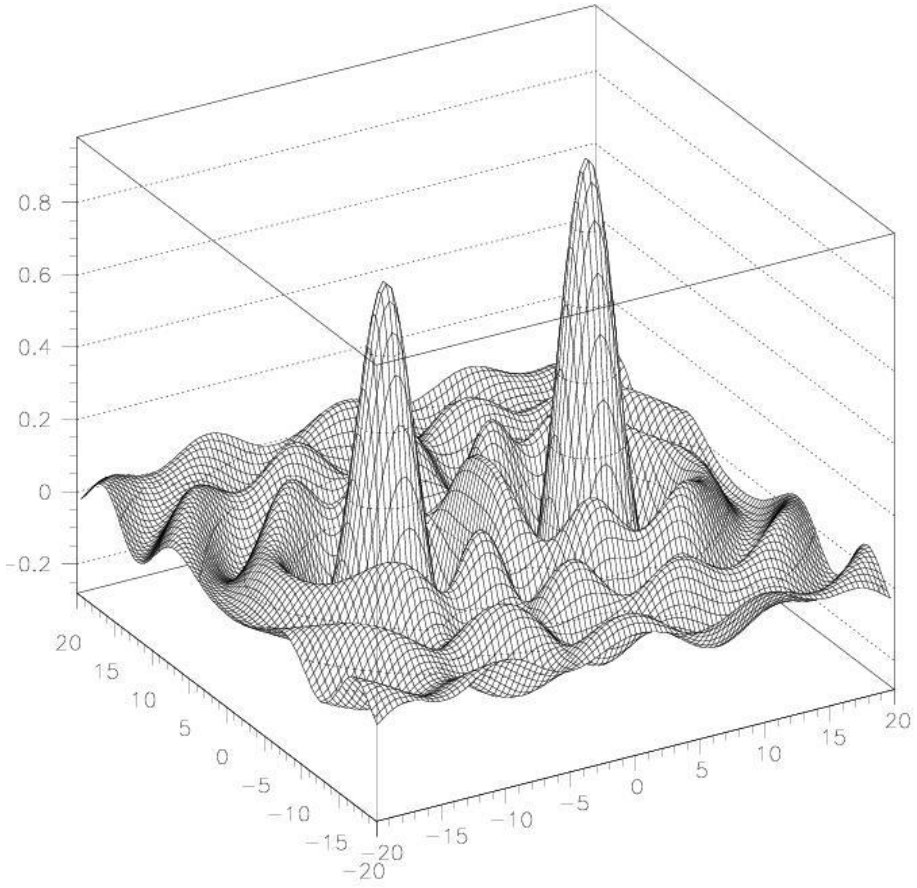
The simple GA

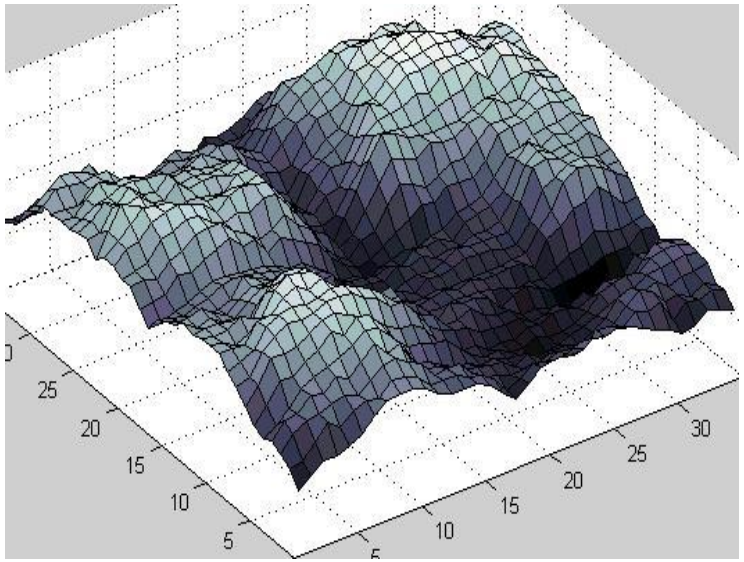
- Has been subject of many (early) studies
  - still often used as benchmark for novel GAs
- Shows many shortcomings, e.g.
  - Representation is too restrictive
  - Mutation & crossovers only applicable for bit-string & integer representations
  - Selection mechanism sensitive for converging populations with close fitness values

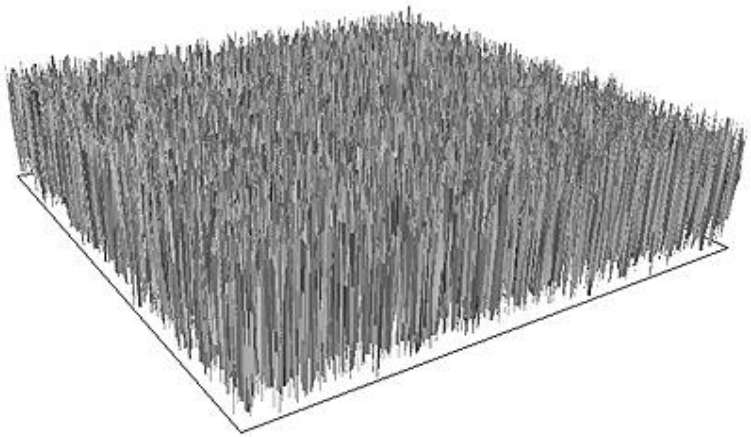
Search Space

- For a simple function f(x) the search space is one dimensional.
- But by encoding several values into the chromosome many dimensions can be searched e.g. two dimensions f(x,y)
- Search space can be visualised as a surface or *fitness landscape* in which fitness dictates height
- Each possible genotype is a point in the space
- A GA tries to move the points to better places (higher fitness) in the space

Fitness landscapes







Search Space

- The nature of the search space dictates how a GA will perform
- A completely random space would be bad for a GA
- Also GA's can get stuck in local maxima if search spaces contain lots of these
- Generally, spaces in which small improvements get closer to the global optimum are good

Issues for GA Practitioners

- Order of genes in chromosome can be important
- Many different coding representations for the parameters of a solution are possible
- Good coding is probably the most important factor for the performance of a GA
- Solution is only as good as the evaluation function (often the hardest part)

Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Support multi-objective optimization
- Always an answer and answer gets better with time
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications.
- Many ways to speed up and improve a GA-based application as knowledge about problem domain is gained
- Good for “noisy” environments

GAs Applications

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

Example : Traveling Salesman Problem

The **Traveling Salesman Problem** is defined as:

'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

- Each city can be represented by an integer .

- Consider the 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 (cont'd)

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

  for example: 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.

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

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

- This can be calculated as the sum of the distances traversed in each travel segment.

The Smaller The Sum, The Better The Solution Represented By That Chromosome.

Fitness Function (cont'd)

- 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 kms.*

- Since our objective is to Minimize the distance, the smaller the total distance, the better the solution.

	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

Selection Operator

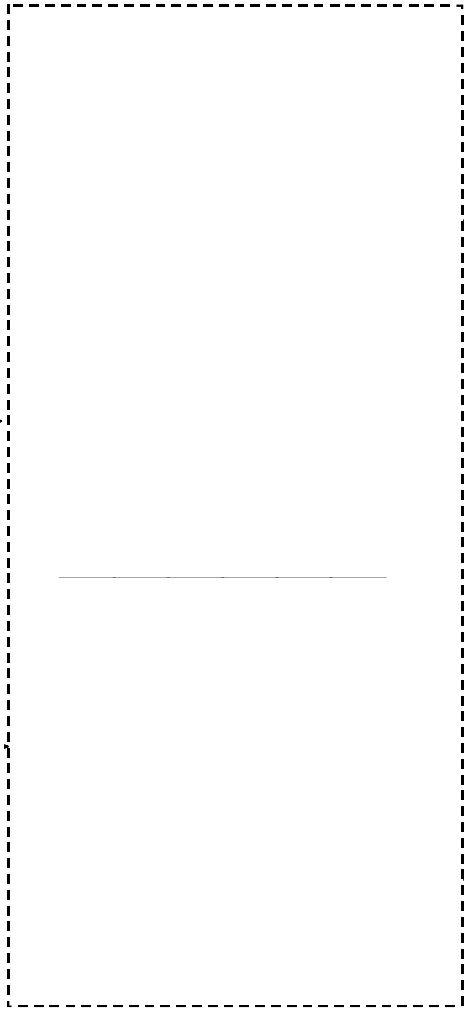
We will use ***Tournament Selection***.

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

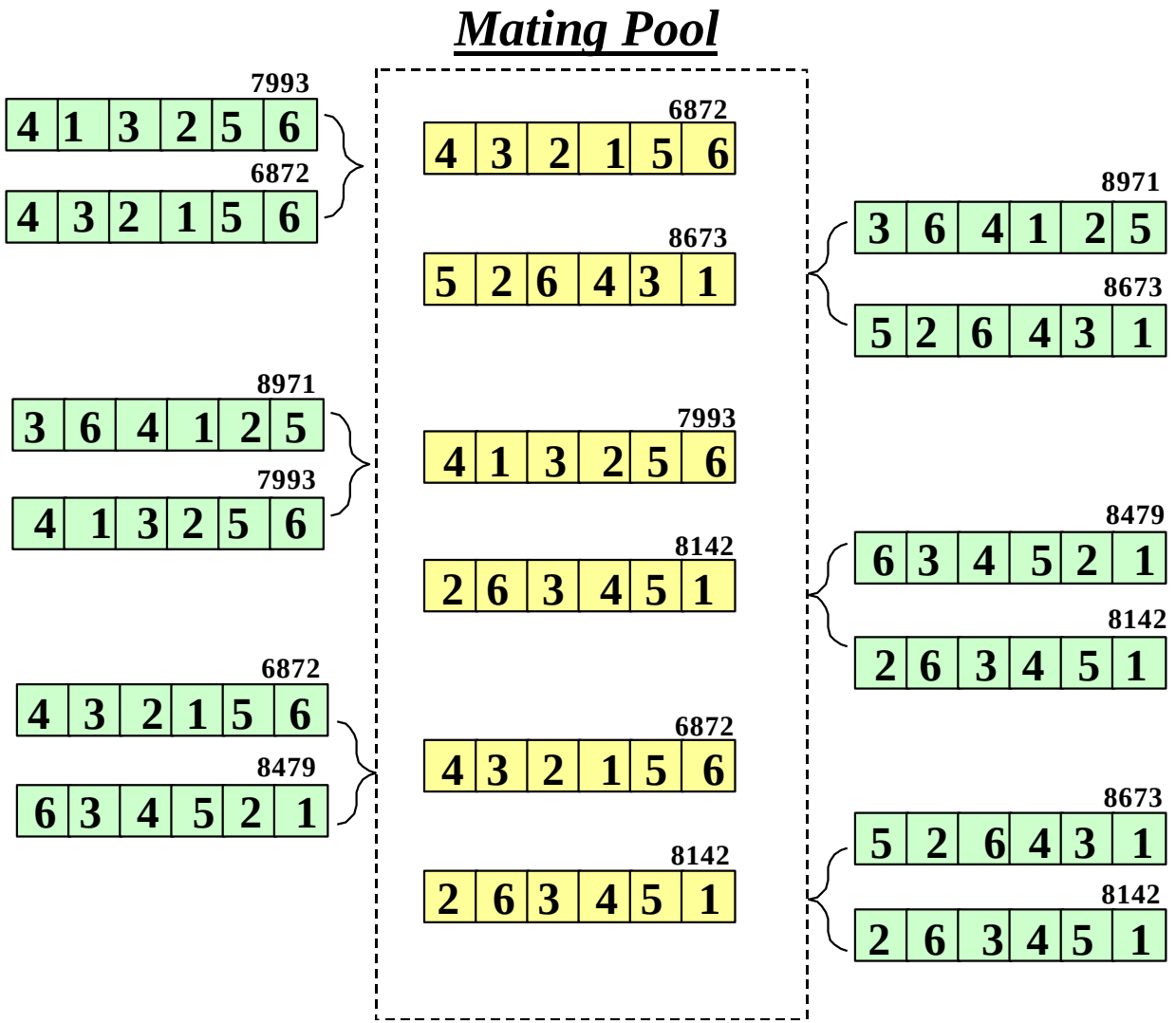
Two other solutions are picked again and another slot in the *mating pool* is filled up with the better solution.

Tournament Selection (cont'd)

Mating Pool

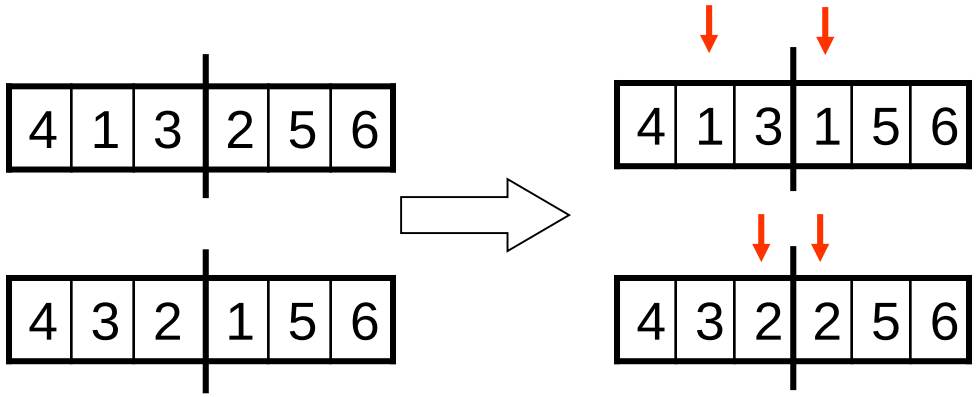


Tournament Selection (cont'd)



Why can’t we 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.

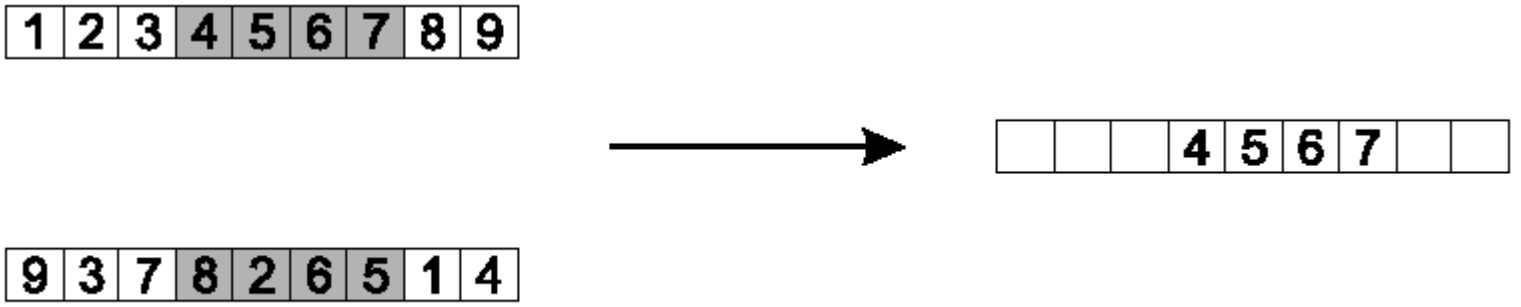


Order-1 crossover

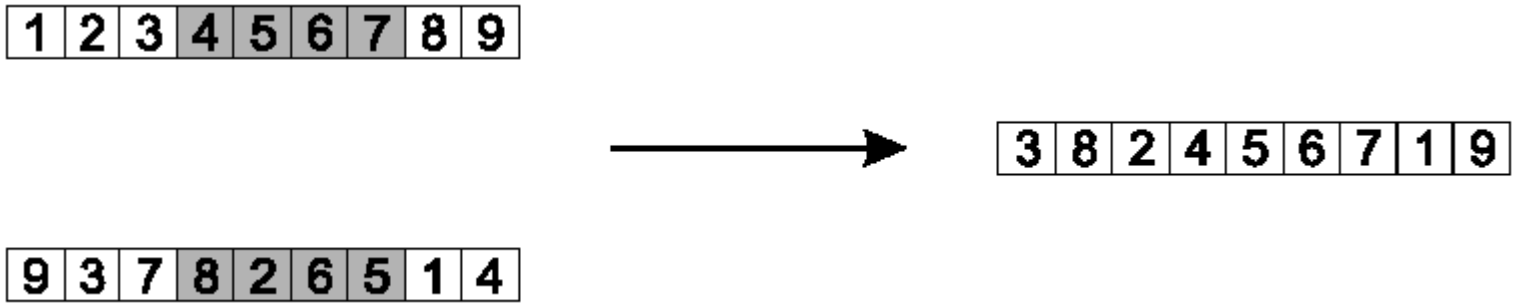
- Idea is to preserve relative order that elements occur
- Informal procedure:
  1. Choose an arbitrary part from the first parent
  2. Copy this part to the first child
  3. Copy the numbers that are not in the first part, to the first child:
     - starting right from cut point of the copied part,
     - using the **order** of the second parent
     - and wrapping around at the end
  4. Analogous for the second child, with parent roles reversed

Order 1 crossover example

- Copy randomly selected set from first parent

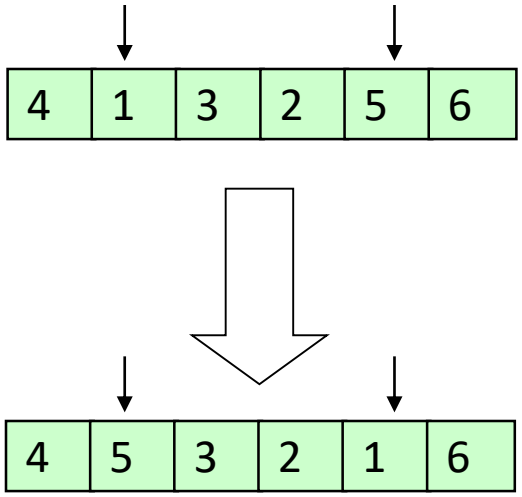


- Copy ***rest*** from second parent in order 1,9,3,8,2

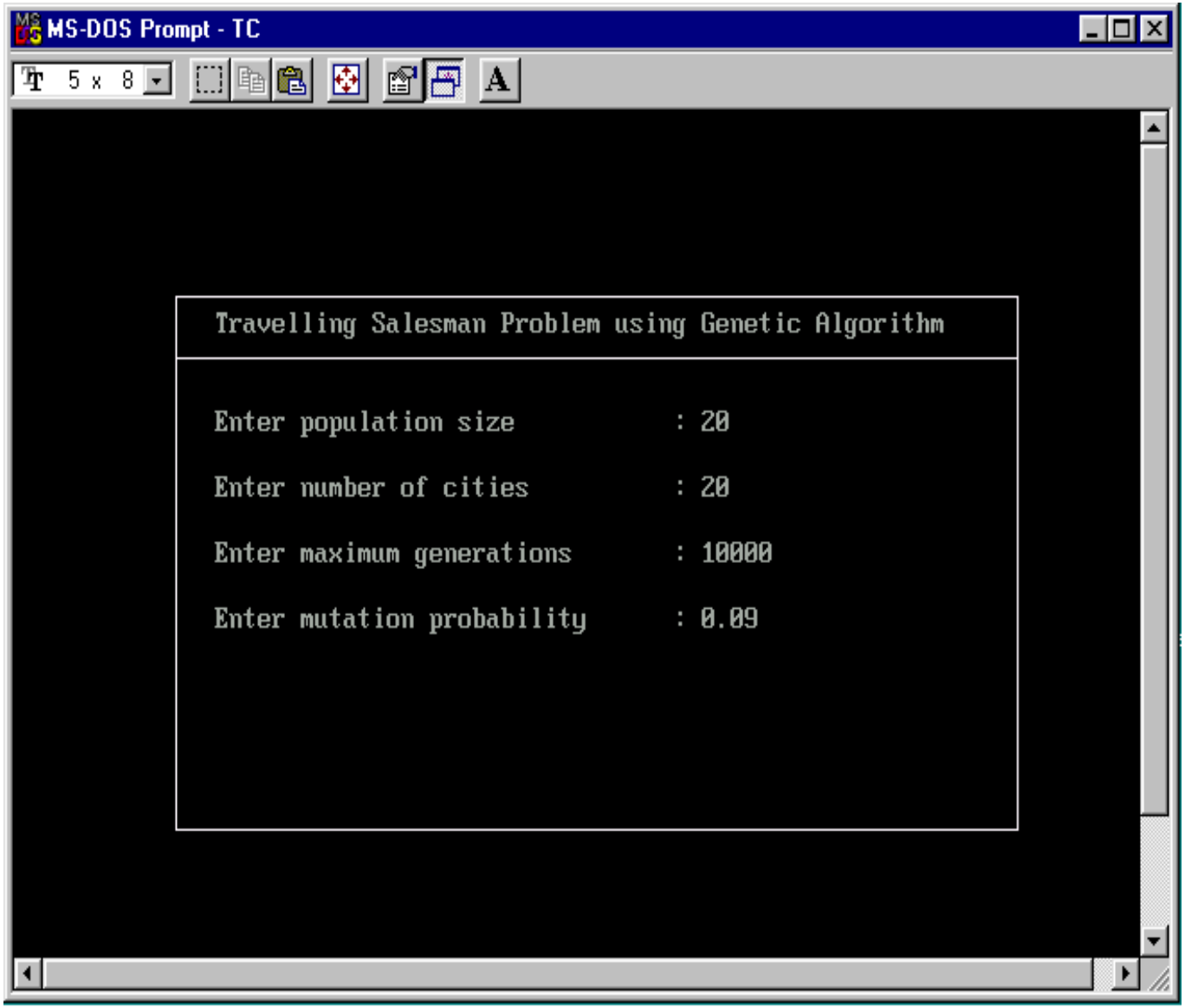


Mutation Operator

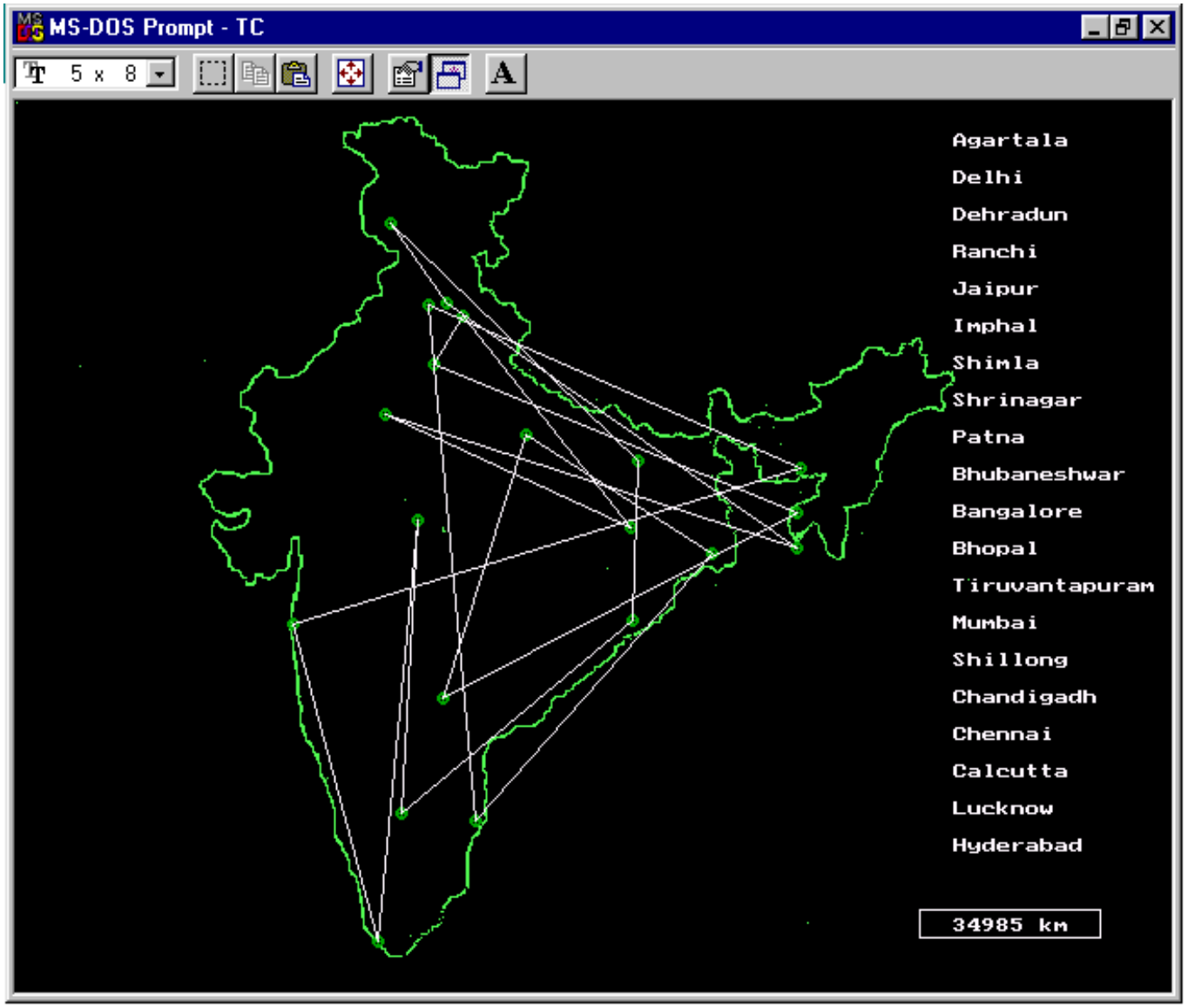
- The mutation operator induces a change in the solution, so as to maintain diversity in the population and prevent *Premature Convergence*.
- Here, we can mutate the string by randomly selecting any two cities and interchanging their positions in the solution, thus giving rise to a new tour.



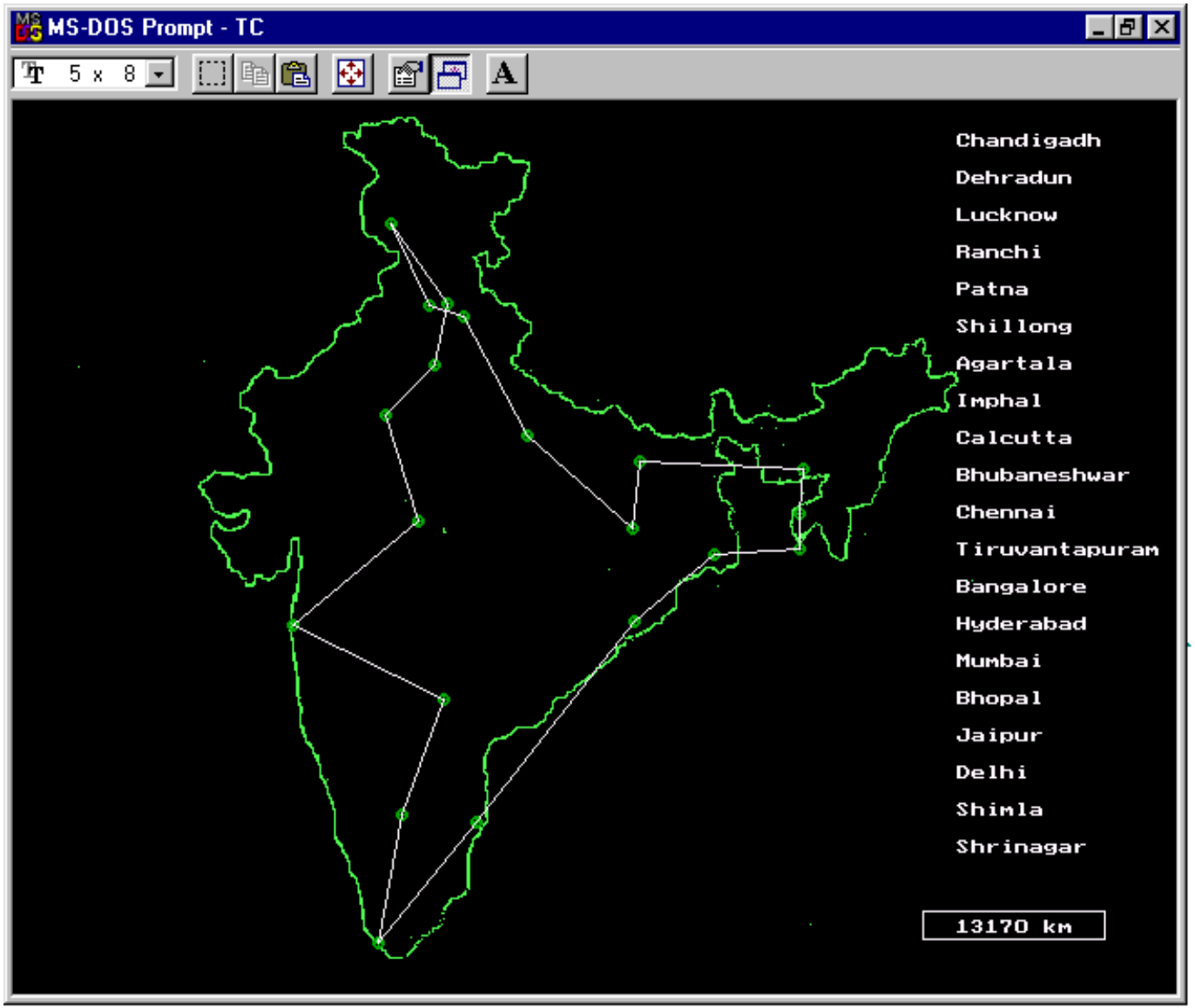
TSP: Input To Program



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

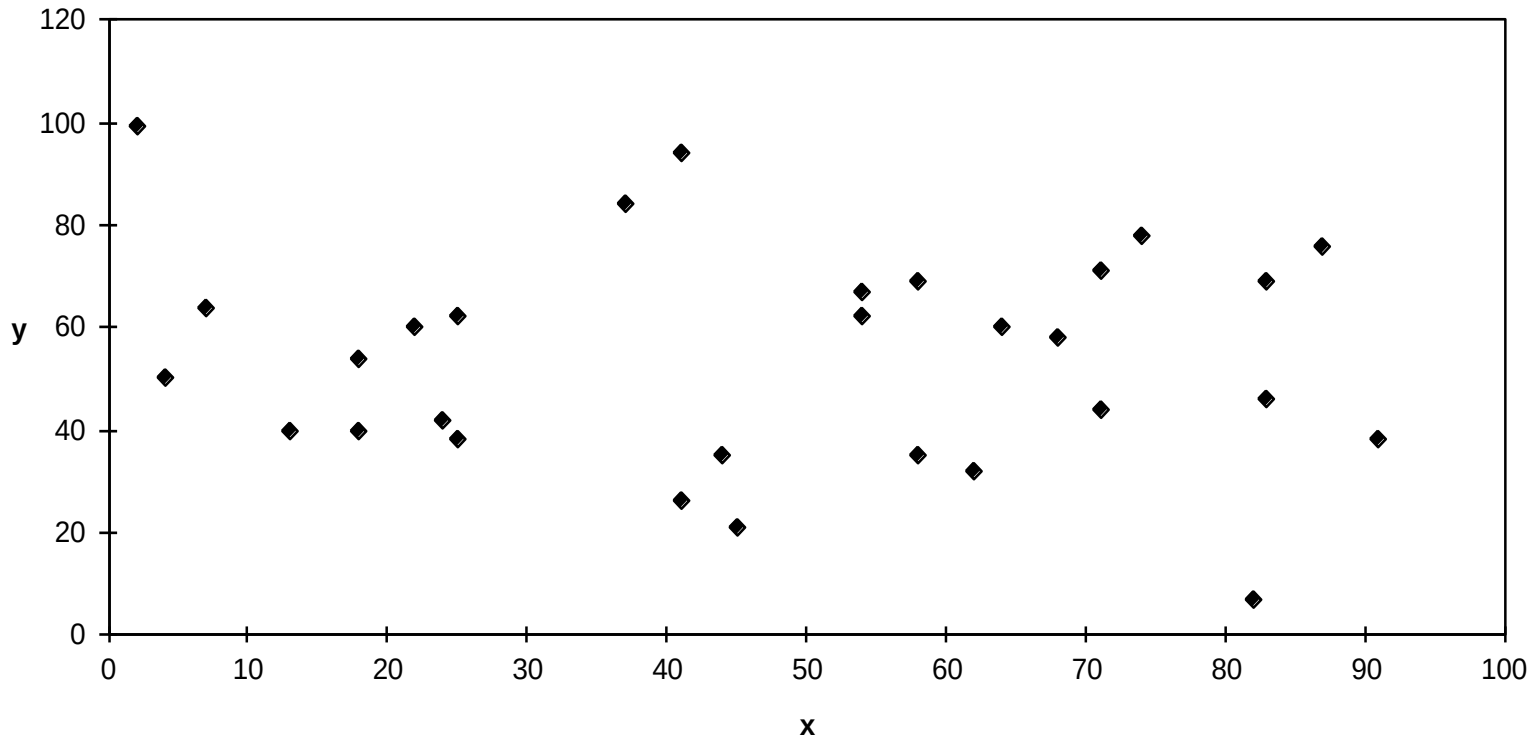


Final Output For 20 cities : Distance=13170 km
Generation 4786

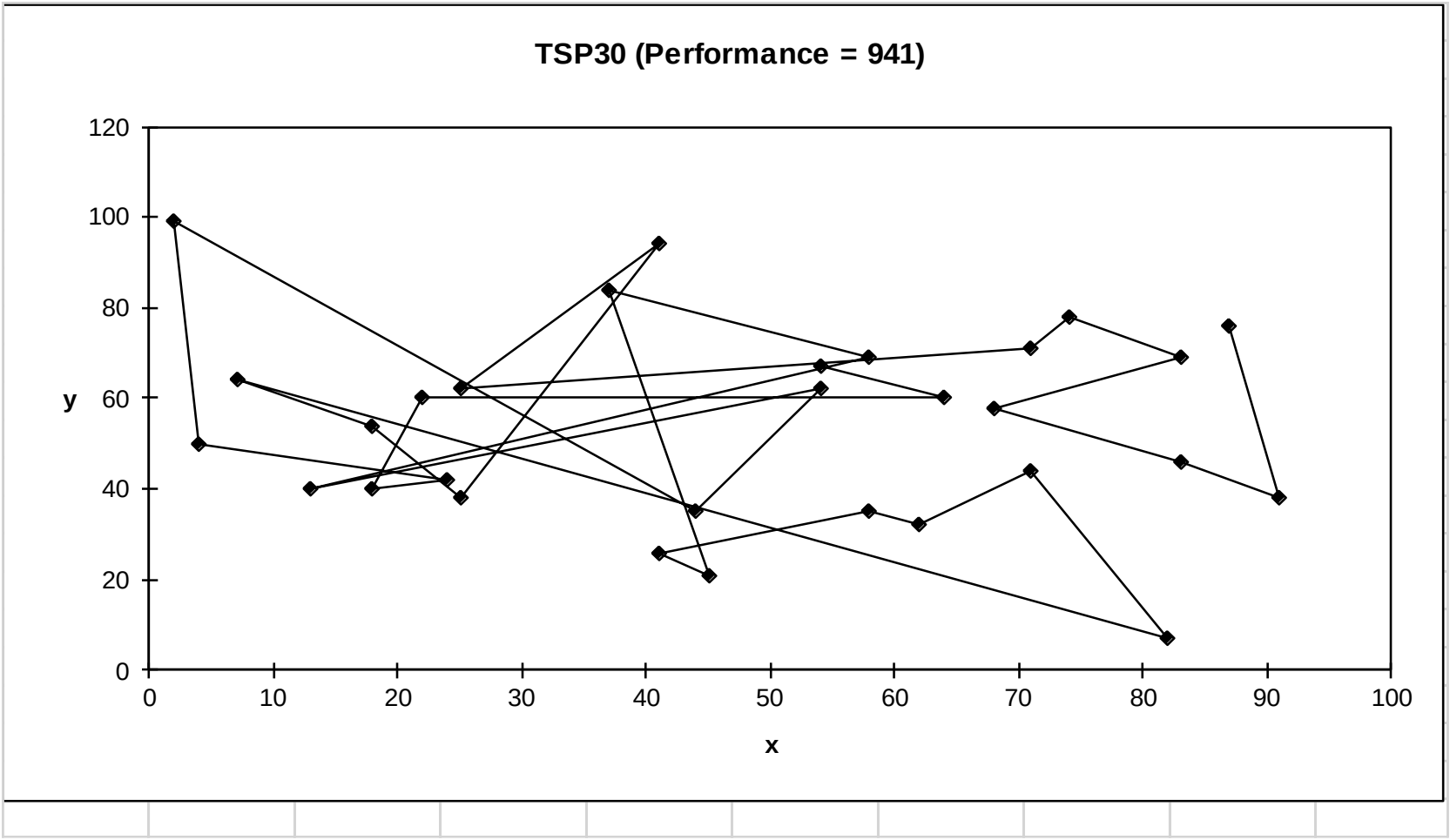


TSP Example: 30 Cities

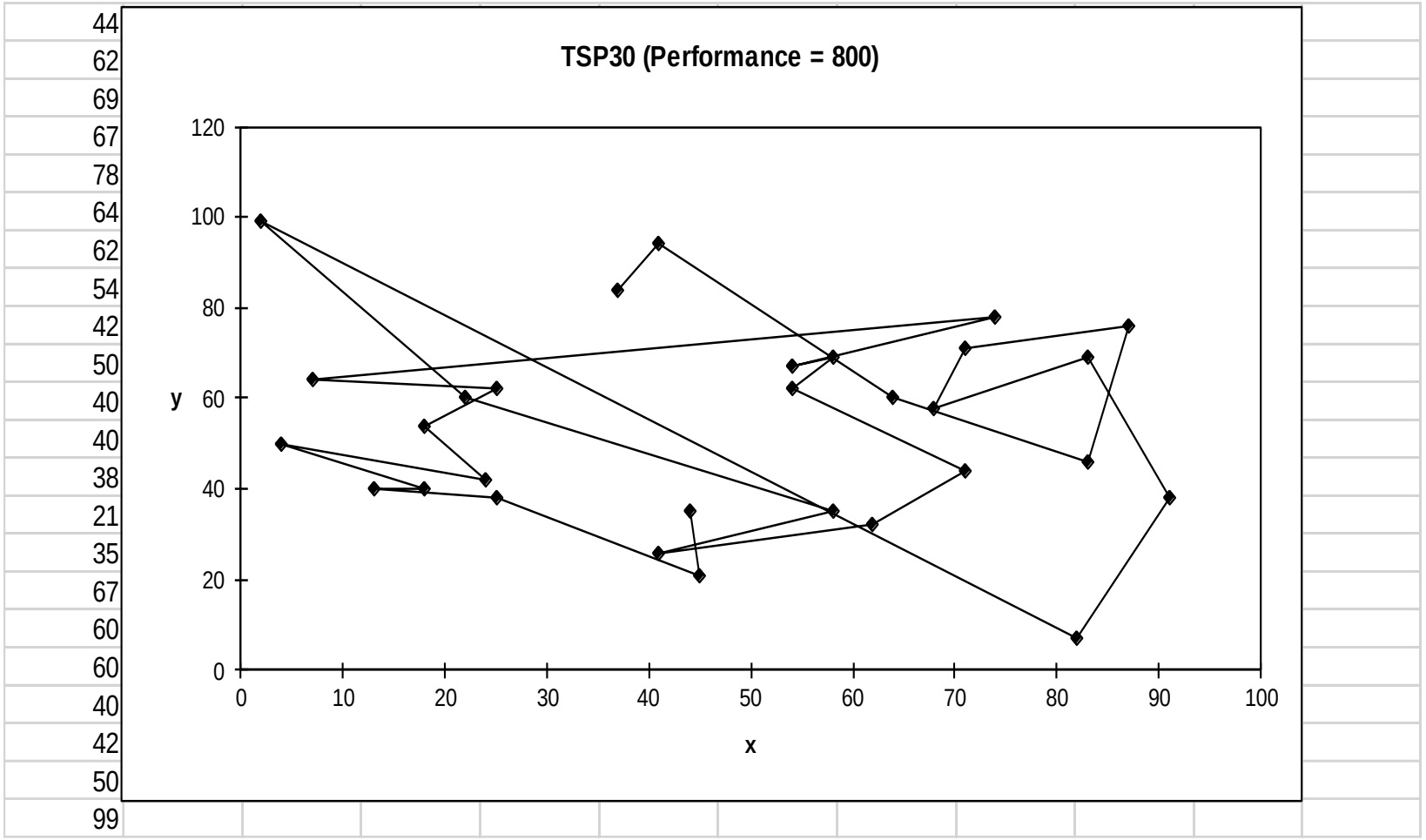
What will be the objective function?



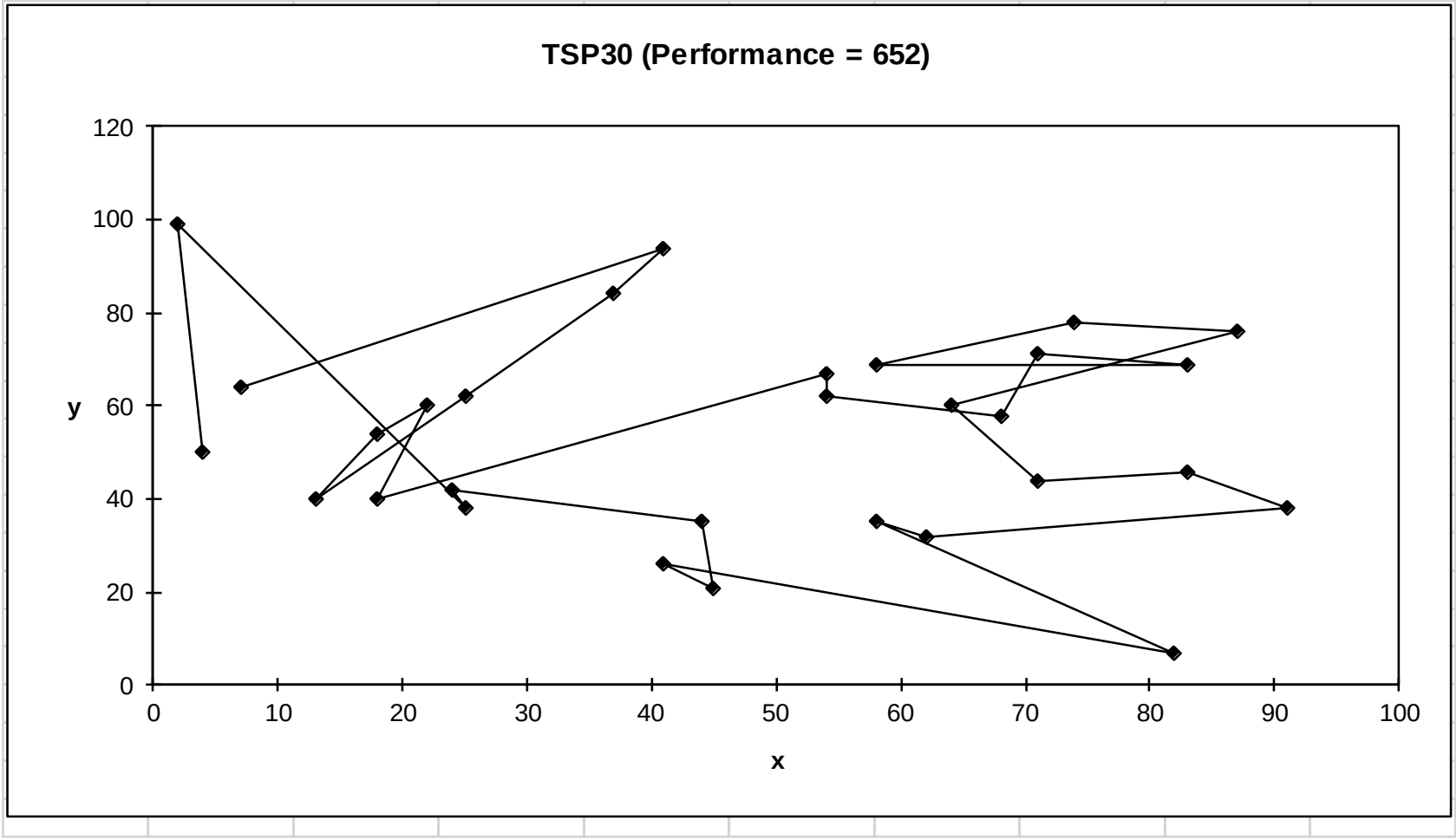
Solution ¡ (Distance = 941)



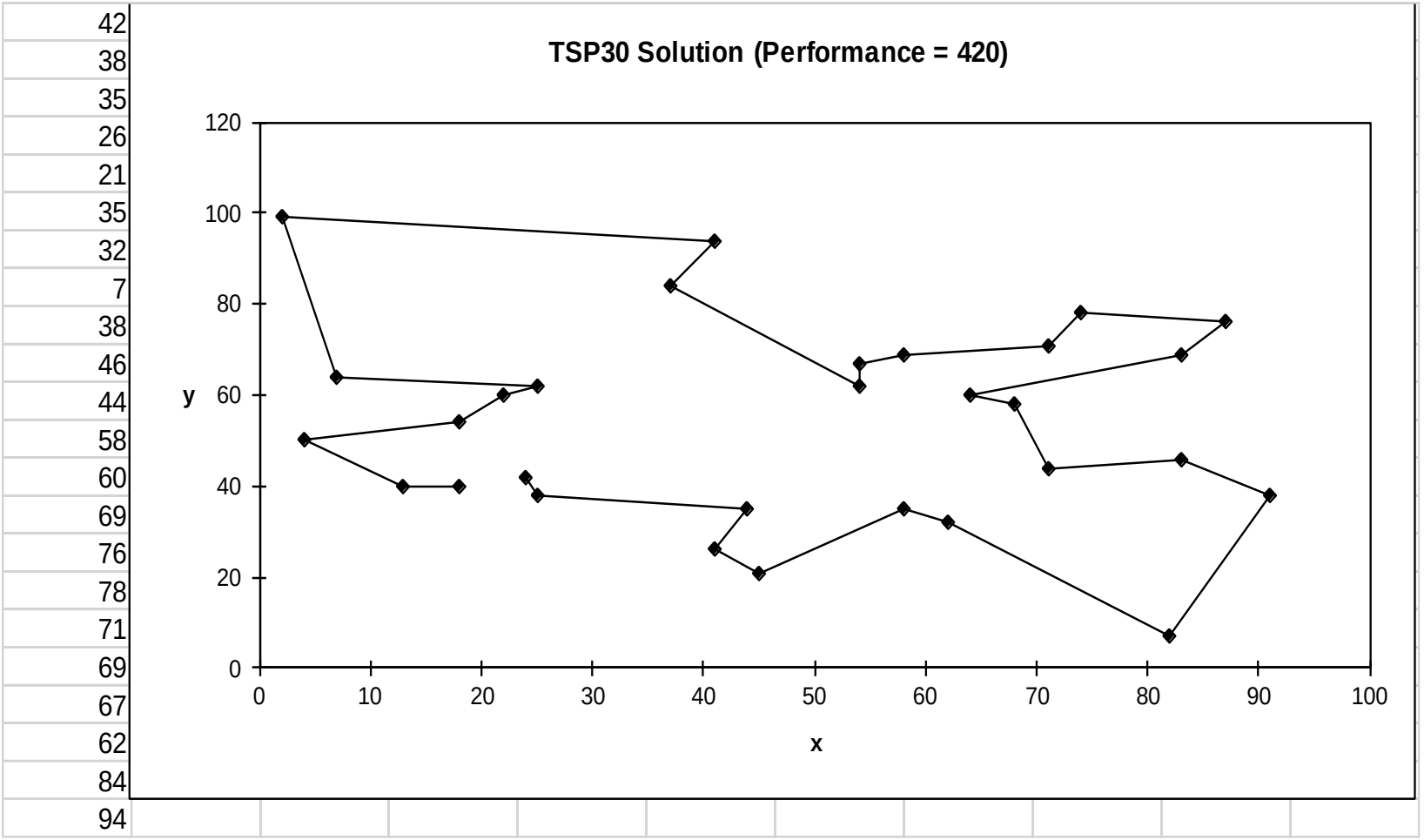
Solution $_j$(Distance = 800)



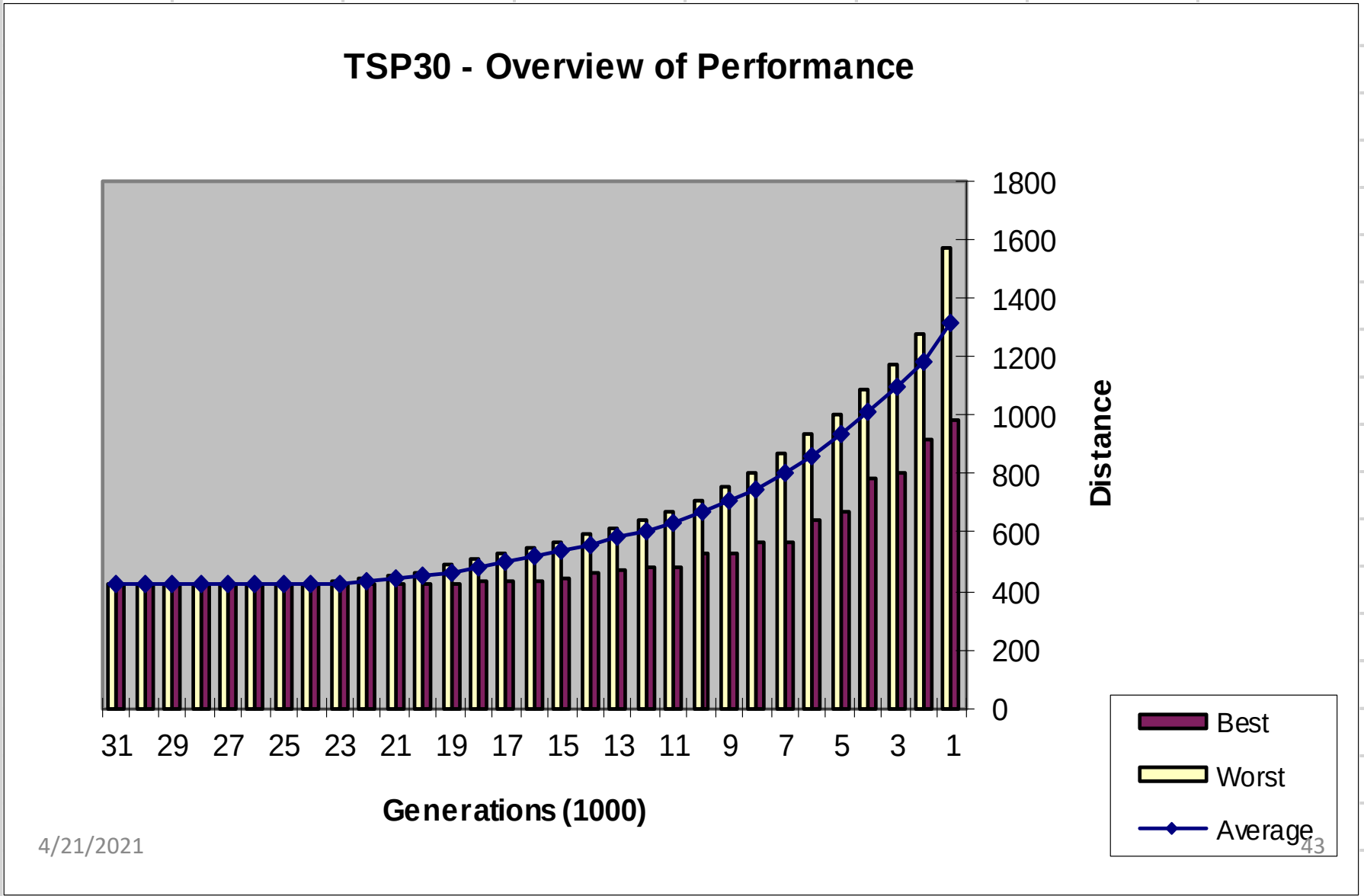
Solution $_k$(Distance = 652)



Best Solution (Distance = 420)



Overview of Performance



Various Strategies for the Genetic Operators

- Different GAs use different …… strategies.
  - Representation (encoding/decoding)
  - Crossover
  - Mutation
  - Selection
  - Replacement