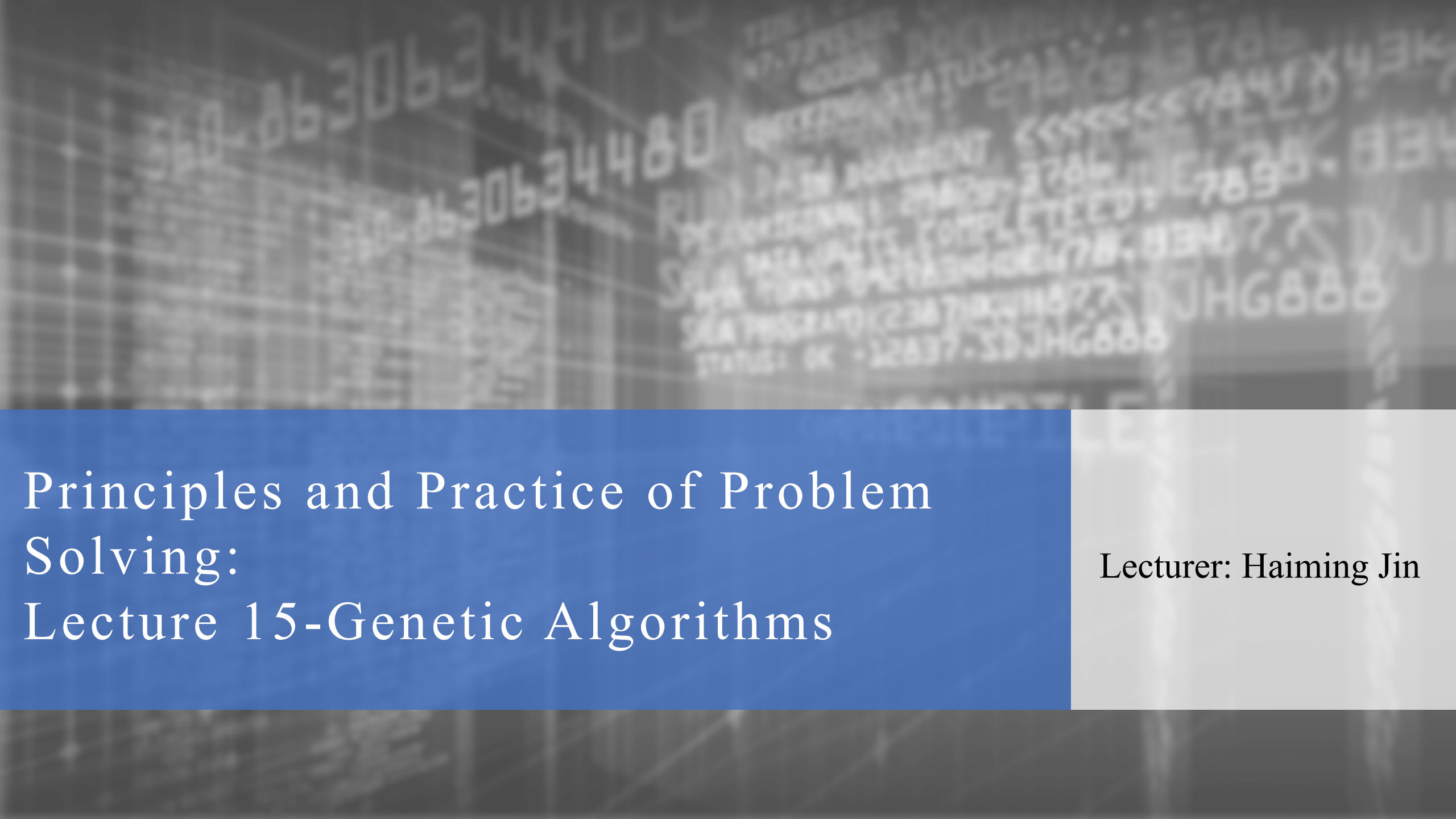


Principles and Practice of Problem Solving: Lecture 15-Genetic Algorithms

Lecturer: Haiming Jin

History of GAs

- As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments.

- By the 1975, the publication of the book ***Adaptation in Natural and Artificial Systems***, by Holland and his students and colleagues.

History of GAs

- early to mid-1980s, genetic algorithms were being applied to a broad range of subjects.

- In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method "genetic programming" (GP).

What is GA ?

- 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 categorized as **global search heuristics**.

- 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**).

What is GA ?

- 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..

What is GA ?

- 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.

No convergence rule or guarantee!

Vocabulary

- **Individual** - Any possible solution
- **Population** - Group of all individuals
- **Fitness** – Target function that we are optimizing (each individual has a fitness)
- **Trait** - Possible aspect (features) of an individual
- **Genome** - Collection of all chromosomes (traits) for an individual

Basic Genetic Algorithm

- Start with a large "population" of randomly generated "attempted solutions" to a problem
  - Repeatedly do the following:
    - Evaluate each of the attempted solutions
    - (probabilistically) keep a subset of the best solutions
    - Use these solutions to generate a new population
- Quit when you have a satisfactory solution (or you run out of time)

The MAXONE Problem

- Suppose we want to maximize the number of ones in a string of l binary digits
- **Is it a trivial problem?**
- 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

The MAXONE Problem

- An individual is encoded (naturally) as a string of l binary digits.
- 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 $n = 6$.

The MAXONE Problem-Initialization

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

$$
\begin{aligned}
s_1 &= 1111010101, & f(s_1) &= 7 \\
s_2 &= 0111000101, & f(s_2) &= 5 \\
s_3 &= 1110110101, & f(s_3) &= 7 \\
s_4 &= 0100010011, & f(s_4) &= 4 \\
s_5 &= 1110111101, & f(s_5) &= 8 \\
s_6 &= 0100110000, & f(s_6) &= 3
\end{aligned}
$$

The MAXONE Problem-Step 1: Selection

- We randomly (using a biased coin) select a subset of the individuals based on their fitness:

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

The MAXONE Problem-Step 1: Selection

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

$$
\begin{aligned}
s_1' &= 1111010101\ (s_1)\\
s_2' &= 1110110101\ (s_3)\\
s_3' &= 1110111101\ (s_5)\\
s_4' &= 0111000101\ (s_2)\\
s_5' &= 0100010011\ (s_4)\\
s_6' &= 1110111101\ (s_5)
\end{aligned}
$$

The MAXONE Problem-Step 2: Crossover

- Next we mate strings for crossover.
  - For each couple we first decide (using some pre-defined probability, for instance 0.6) whether to actually perform the crossover or not.
  - If we decide to actually perform crossover, we randomly extract the crossover points, for instance 2 and 5.

The MAXONE Problem-Step 2: Crossover

- Before crossover:
  - $s_1' = 11\mathbf{110}10101$, $s_2' = 11\mathbf{101}10101$
- After crossover:
  - $s_1'' = 11\mathbf{101}10101$, $s_2'' = 11\mathbf{110}10101$

The MAXONE Problem-Step 3: Mutations

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

Initial strings	After mutating
$s_1'' = 1110110101$	$s_1''' = 1110100101$
$s_2'' = 1111010101$	$s_2''' = 1111110100$
$s_3'' = 1110111101$	$s_3''' = 1110101111$
$s_4'' = 0111000101$	$s_4''' = 0111000101$
$s_5'' = 0100011101$	$s_5''' = 0100011101$
$s_6'' = 1110110011$	$s_6''' = 1110110001$

And now, iterate …

- 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.

GA Operators

- Methods of **representation**
- Methods of **selection**
- Methods of **reproduction**

Common Representation Methods

- Binary strings
- Arrays of integers (usually bound)
- Arrays of letters

Methods of Selection

- There are many different strategies to select the individuals to be copied over into the next generation.
  - Roulette-wheel selection.
  - Elitist selection.
  - Fitness-proportionate selection.
  - Scaling selection.
  - Rank selection.
  - …

Roulette-Wheel Selection

- Conceptually, this can be represented as a game of roulette - each individual gets a slice of the wheel, but more fit ones get larger slices than less fit ones.

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

Other Selection Methods

- Elitist selection: Chose only the most fit members of each generation.
- Cutoff selection: Select only those that are above a certain cutoff for the target function.

Methods of Reproduction

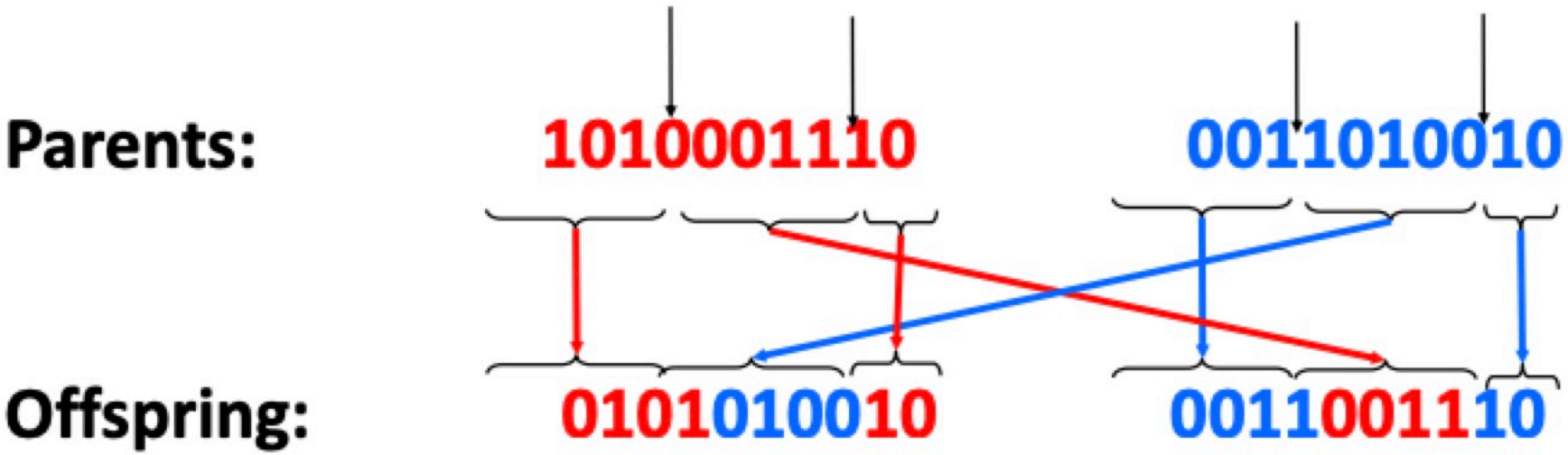
- There are primary methods
  - Crossover
  - Mutation

Methods of Reproduction: Crossover

- Two parents produce two offspring
- Two options:
  - The chromosomes of the two parents are copied to the next generation
  - The two parents are randomly recombined (crossed-over) to form new offsprings
- Possible crossover strategies
  - Randomly select a single point for a crossover
  - Multi-point crossover
  - Uniform crossover

Two-Point Crossover

- Avoids cases where genes at the beginning and end of a chromosome are always split



Uniform Crossover

- A random subset is chosen
- The subset is taken from parent 1 and the other bits from parent 2.

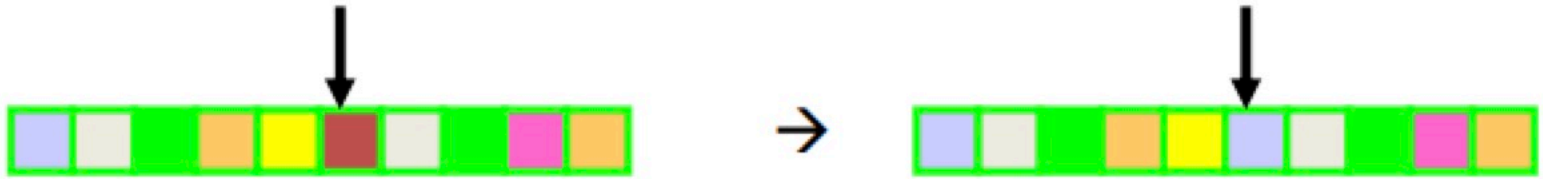
Subset: **BAABBAABBB** **(Randomly generated)**

Parents: **1010001110** **0011010010**

Offspring: **0011001010** **1010010110**

Methods of Reproduction: Mutations

- Generating new offspring from single parent



Example: Traveling Salesman Problem

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

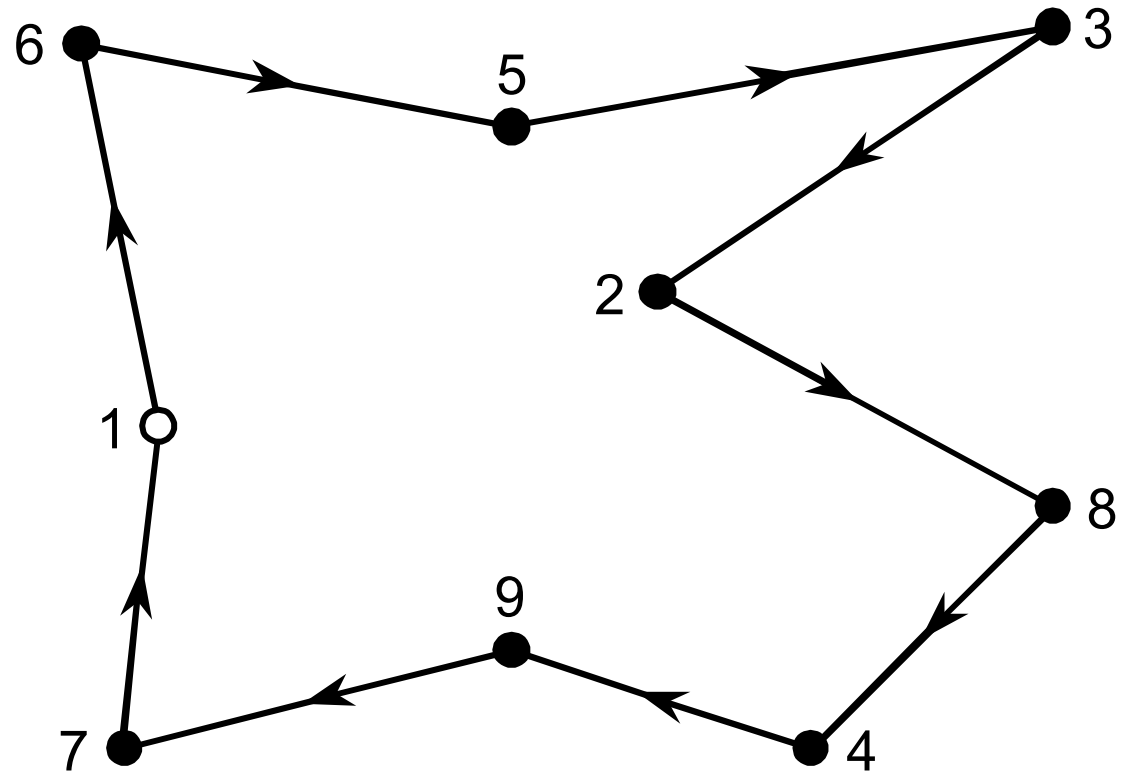
- The TSP is represented in numerous transportation and logistics applications such as
  - arranging routes for school buses to pick up children in a school district,
  - delivering meals to home-bound people,
  - scheduling stacker cranes in a warehouse,
  - planning truck routes to pick up parcel post
  - and many others.

How does a genetic algorithm solve the travelling salesman problem?

- First, we need to decide how to represent a route of the salesman.
  - The most natural way of representing a route is the ***path representation***.
  - Each city is given an alphabetic or numerical name, the route through the cities is represented as a chromosome.
  - Appropriate genetic operators are used to create new routes.

- Suppose we have nine cities named from 1 to 9.
  - In a chromosome, the order of the integers represents the order in which the cities will be visited by the salesman.

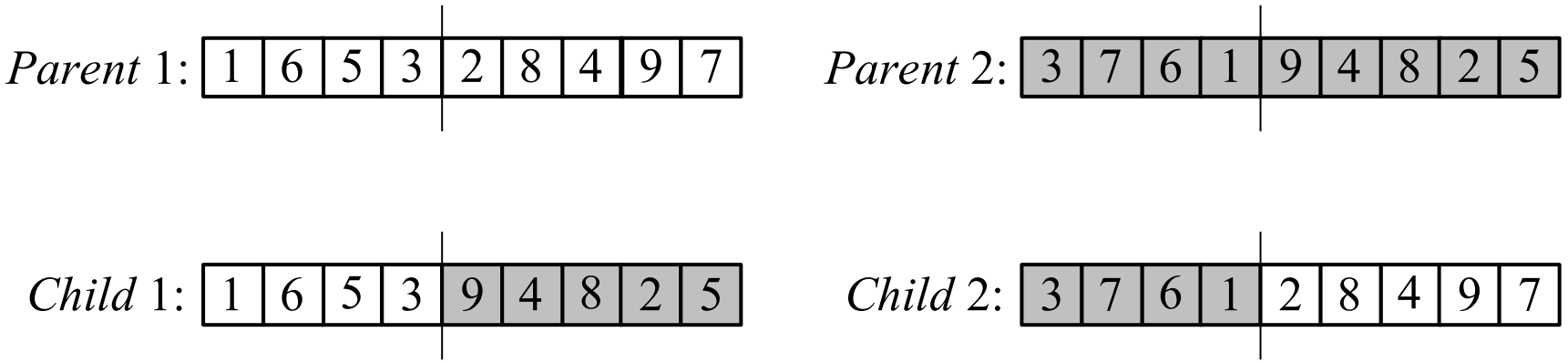
An example of the salesman's route



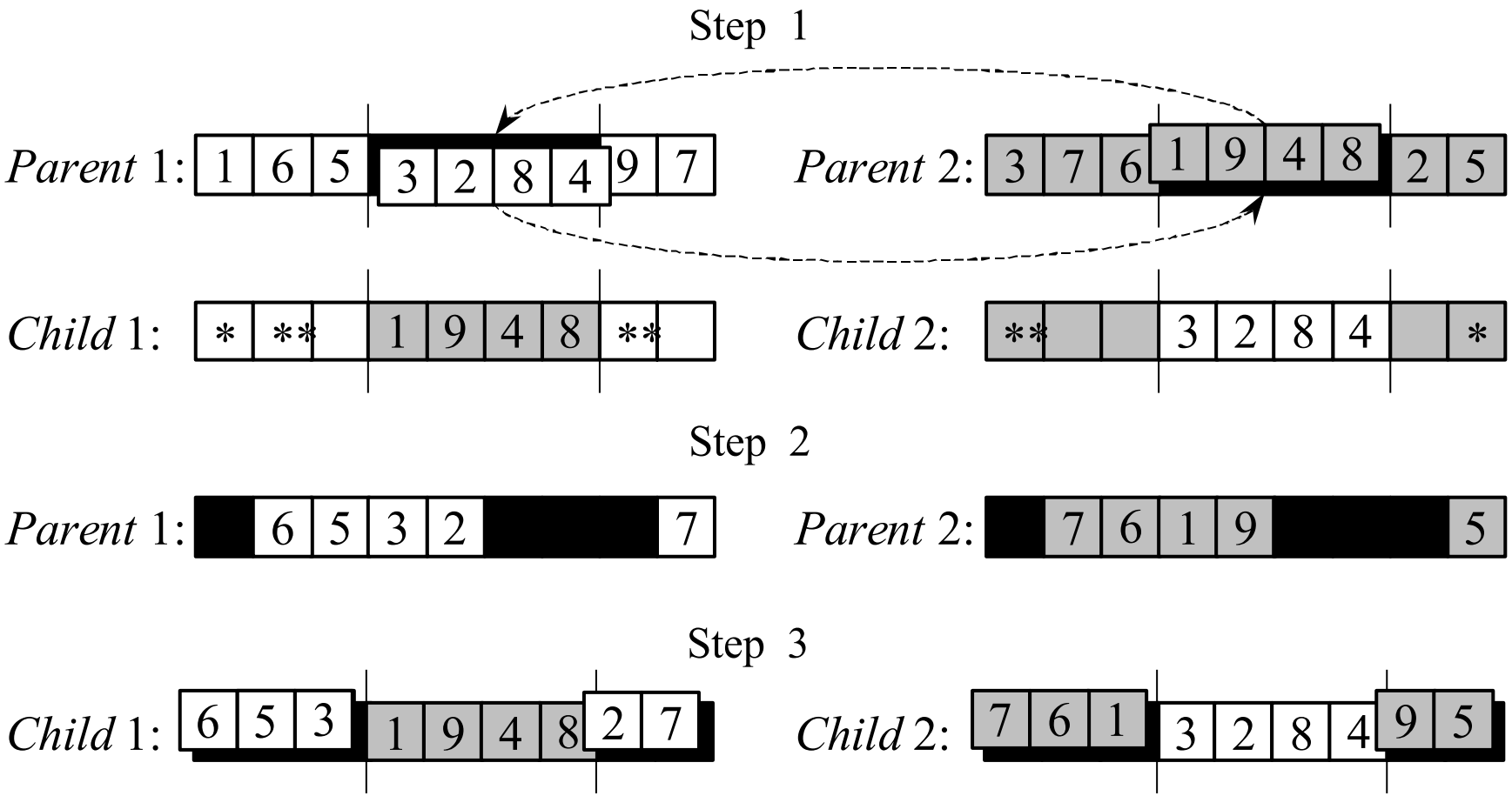
1	6	5	3	2	8	4	9	7

How does the crossover operator works?

- The crossover operator in its classical form cannot be directly applied to the 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:



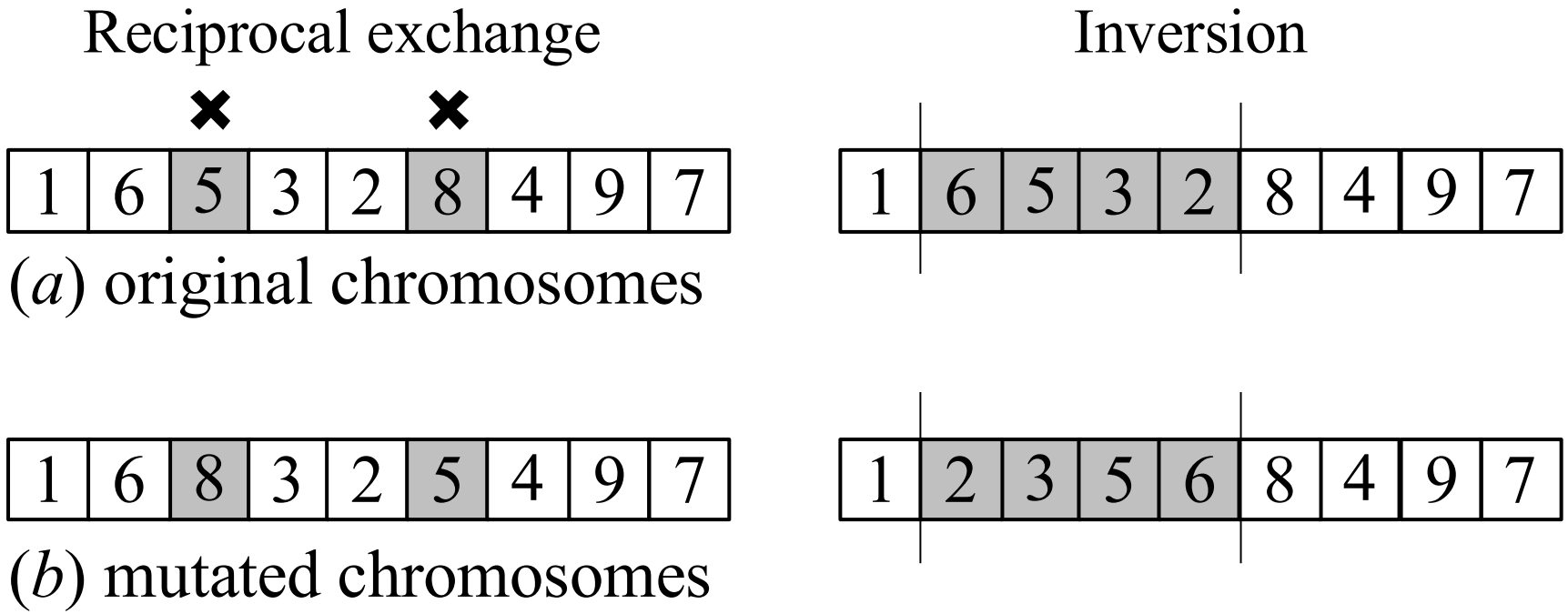
Crossover operators for the TSP



How does the mutation operator works?

- There are two types of mutation operators: **reciprocal exchange** and **inversion**.
  - The reciprocal exchange operator simply swaps two randomly selected cities in the chromosome.
  - The inversion operator selects two random points along the chromosome string and reverses the order of the cities between these points.

Mutation operators for the TSP

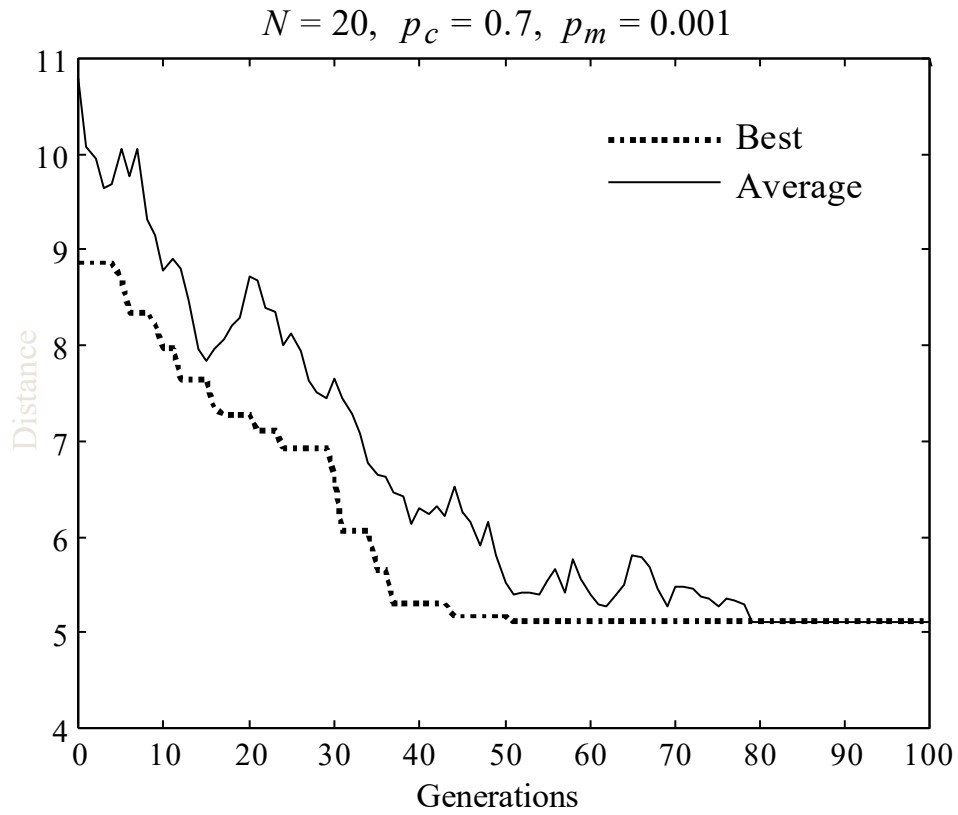


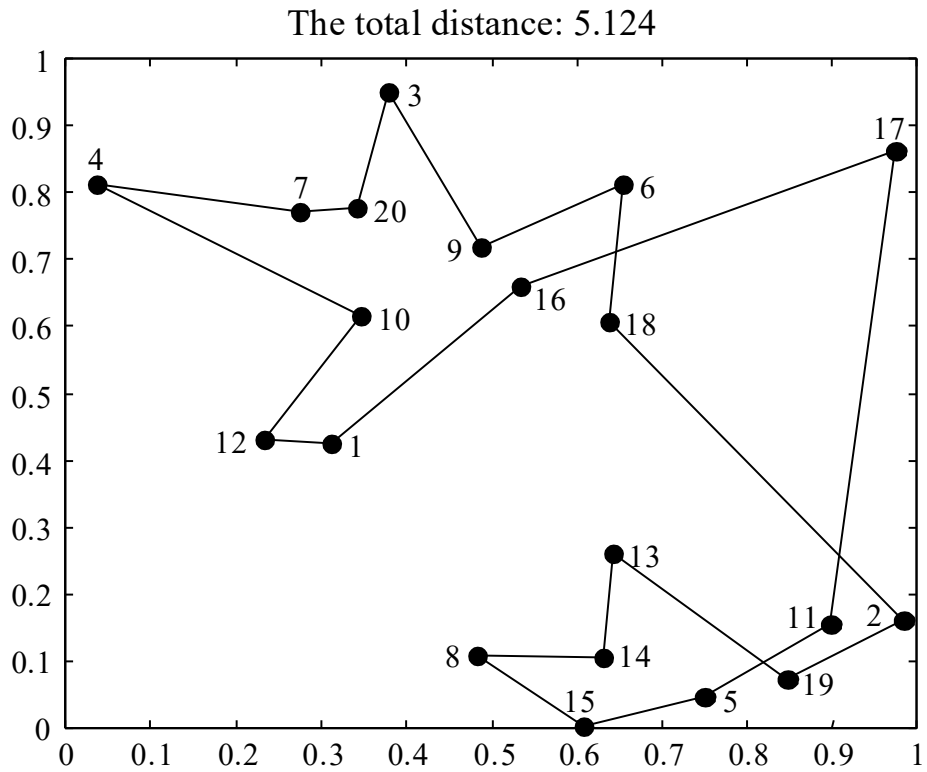
How do we define a fitness function in the TSP?

- The fitness of each individual chromosome is determined as the reciprocal of the route length.

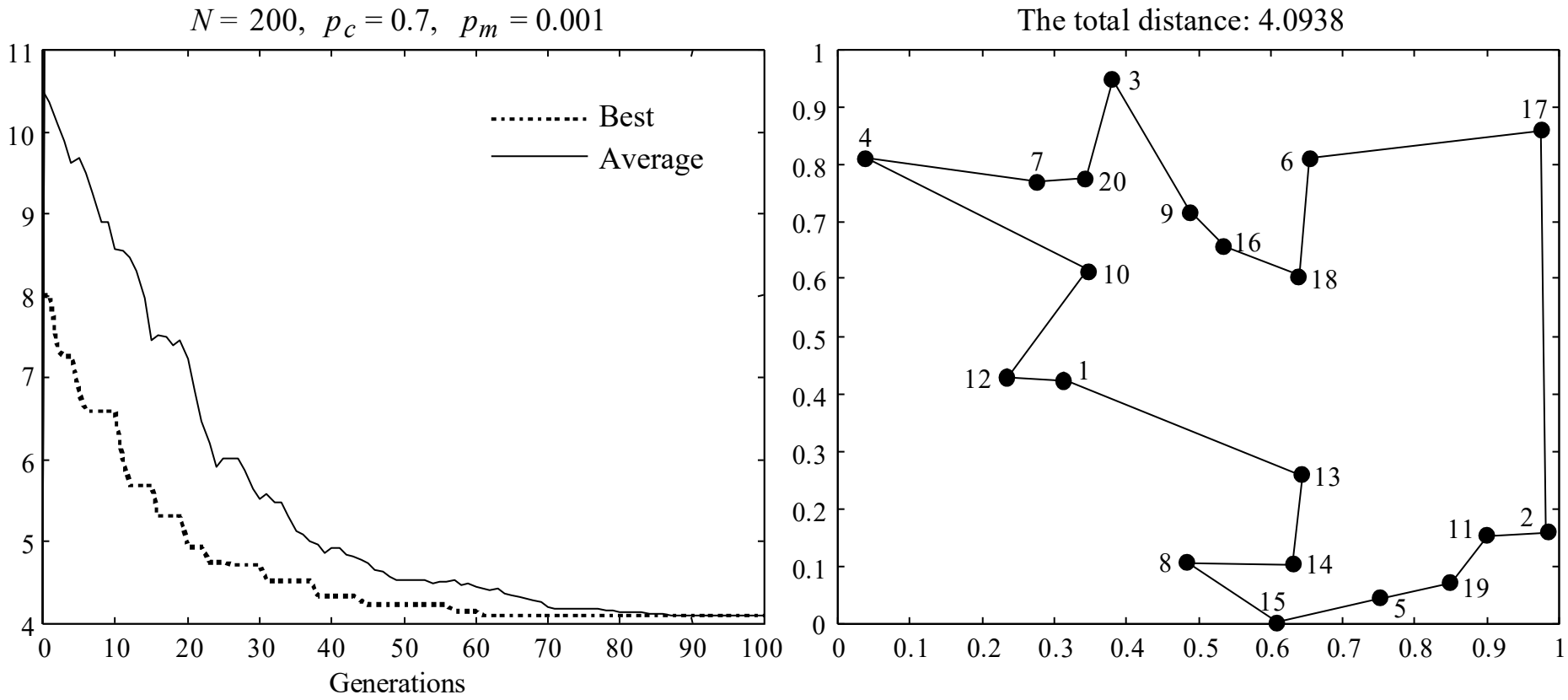
- In other words, the shorter the route, the fitter the chromosome.

Performance graph and the best salesman's route created in a population of 20 chromosomes after 100 generations

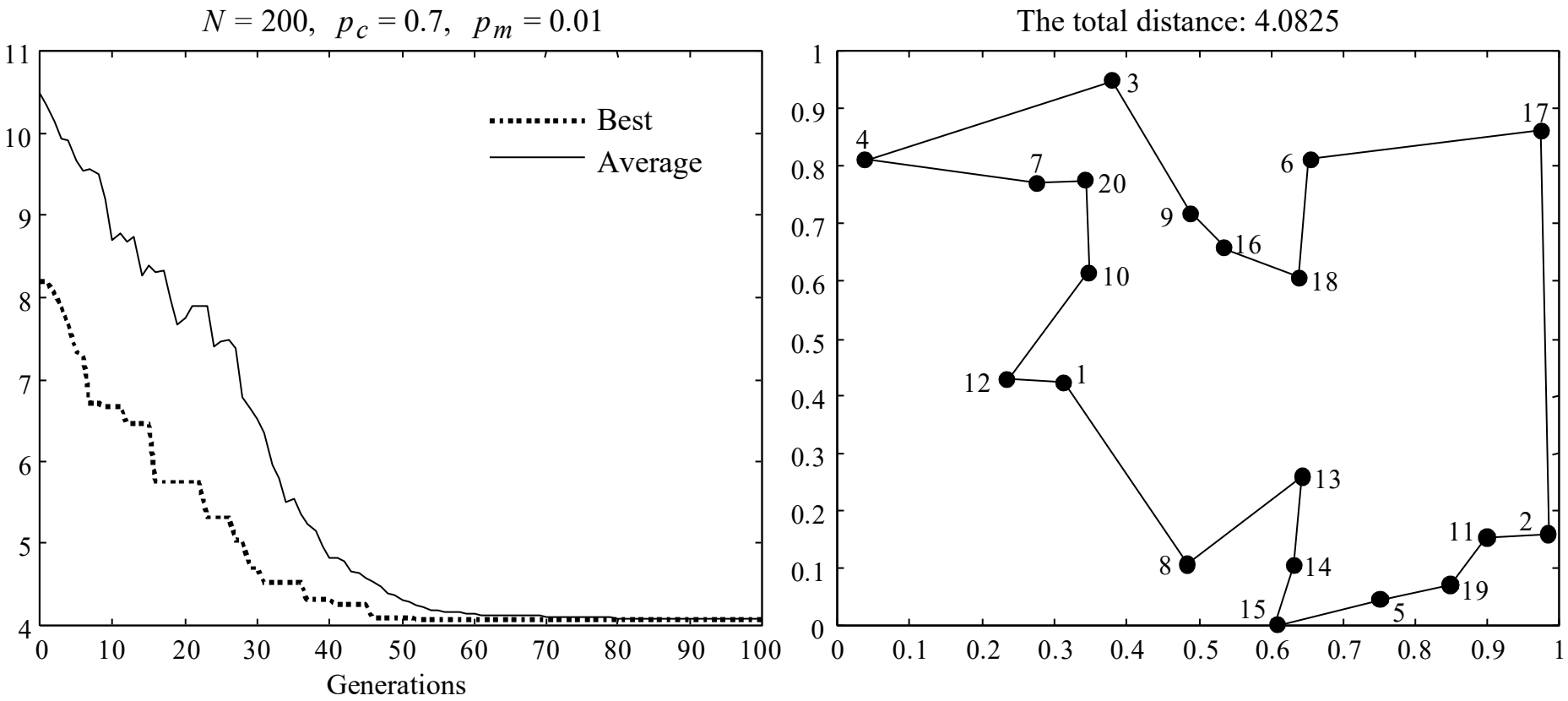




Performance graphs and the best routes created in a population of 200 chromosomes: mutation rate is 0.001



Performance graphs and the best routes created in a population of 200 chromosomes: mutation rate is 0.01



Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Always an answer; answer gets better with time.
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications.