ST. FRANCIS XAVIER
UNIVERSITY

CSCI-564
CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 19 – SELECTIVE SEARCH (CONT'D)

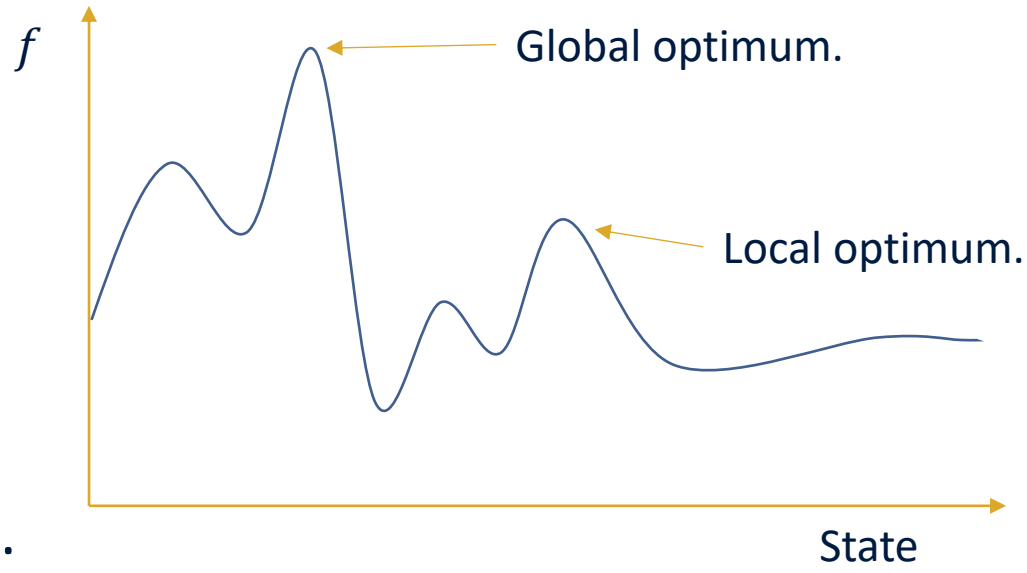
Dr. Jean-Alexis Delamer

Recap

- Some problems can't be solved optimally.
  - But we can find an approximate solution.

- Local search:
  - Consider a local neighborhood.
  - Try to find a global optimum.
  - But we can be stuck in local optimum.

- Some algorithms try to reduce this downside.
  - Hill-Climbing
  - Simulated Annealing



Genetic Algorithms

- Local search techniques are still regular search algorithms.
  - Define successors of a state (neighborhood) differently.
  - Use randomization to avoid local optimum.

- Researchers tried to find other methods to solves hard problems.
  - A very popular methods is genetic algorithms.
  - Only an introduction.

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).

Genetic Algorithms

- What is a genetic algorithm?
  - A genetic algorithm (or GA) is a search technique used to find true or approximate solutions to optimization and search problems.
  - They are global search heuristics.
  - They are a class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).

Genetic Algorithms

- How does it work?
  - 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.
  - Terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Genetic Algorithms

- Terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.
- Why?
  - No convergence rule or guarantee.
  - The number of generation s to make sure it terminates.

Genetic Algorithms

- We need to define some terms first.
  - Individual: Any possible solutions.
  - Population: Group of all individuals.
  - Fitness: Target function that we are optimizing, our f-value.
  - Trait: Possible aspect (features) of an individual.
  - Genome: Collection of all chromosomes (traits) for an individual.

Genetic Algorithms

- Basic Genetic Algorithm:
  - Start with a large "population" randomly generated.
    - Each individual is a potential solution to a problem
  - Repeatedly do the following:
    - Evaluate each individual.
    - Probabilistically keep a subset of the best solutions.
    - Use these solutions to generate a new population.
  - Stop when you reach a terminate condition (good solution or number of generation).

Genetic Algorithms

- Example MaxOne problem:
  - Suppose we want to maximize the number of ones in a string of n binary digits.
  - An individual is encoded as a string of n binary digits.
  - The fitness f of a candidate solution to he MaxOne problem is the number of ones in its genetic code.
  - We start with a population of k random strings (individuals).

Genetic Algorithms

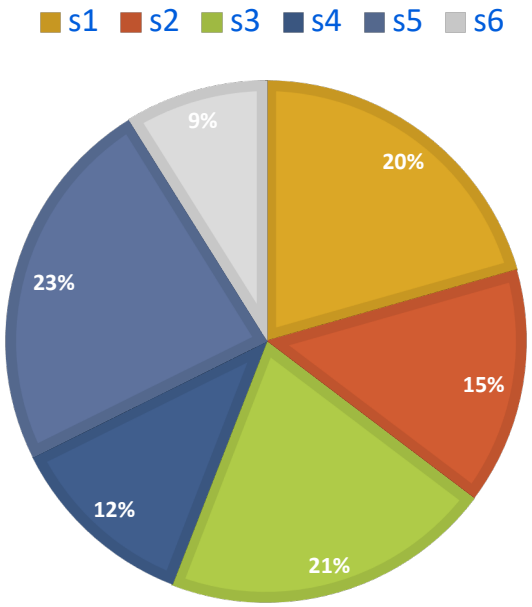
- Example:
  - We suppose that $n = 10$ and $k = 6$.
  - We generate a first population:

$$s_1 = 1111010101\ f(s_1) = 7$$
$$s_2 = 0111000101\ f(s_2) = 5$$
$$s_3 = 1110110101\ f(s_3) = 7$$
$$s_4 = 0100010011\ f(s_4) = 4$$
$$s_5 = 1110111101\ f(s_5) = 8$$
$$s_6 = 0100110000\ f(s_6) = 3$$

Genetic Algorithms

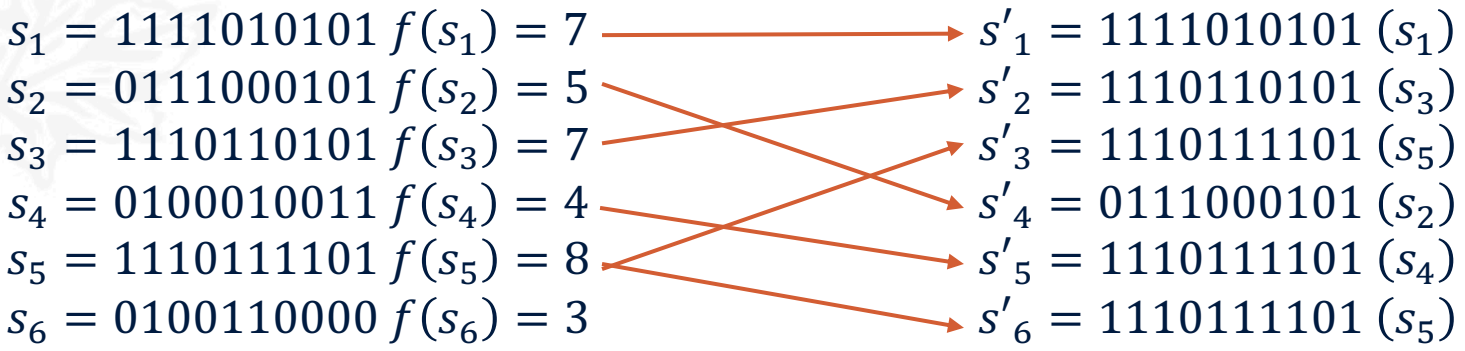
- Step 1: Selection
  - We randomly (using a biased coin) select a subset of the individuals based on their fitness:
    - Individual i will have a probability to be chosen $\frac{f(i)}{\sum_i f(i)}$

$$s_1 = 1111010101\ f(s_1) = 7$$
$$s_2 = 0111000101\ f(s_2) = 5$$
$$s_3 = 1110110101\ f(s_3) = 7$$
$$s_4 = 0100010011\ f(s_4) = 4$$
$$s_5 = 1110111101\ f(s_5) = 8$$
$$s_6 = 0100110000\ f(s_6) = 3$$



Genetic Algorithms

- Example:
  - Suppose that after performing we get the following:



Genetic Algorithms

- 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 perform the crossover or not.
  - If we decide to perform crossover, we randomly extract the crossover points.

- Before crossover:

- After crossover:

$$s'_1 = 11110\,10101 \quad s'_2 = 11101\,10101$$

$$s'_1 = 11101\,10101 \quad s'_2 = 11110\,10101$$

Genetic Algorithms

- 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
$s''_1 = 1110110101$	$s''_1 = 1110100101$
$s''_2 = 1111010101$	$s''_2 = 1111110100$
$s''_3 = 1110111101$ →	$s''_3 = 1110101111$
$s''_4 = 0111000101$	$s''_4 = 0111000101$
$s''_5 = 0100011101$	$s''_5 = 0100011101$
$s''_6 = 1110110011$	$s''_6 = 1110110001$

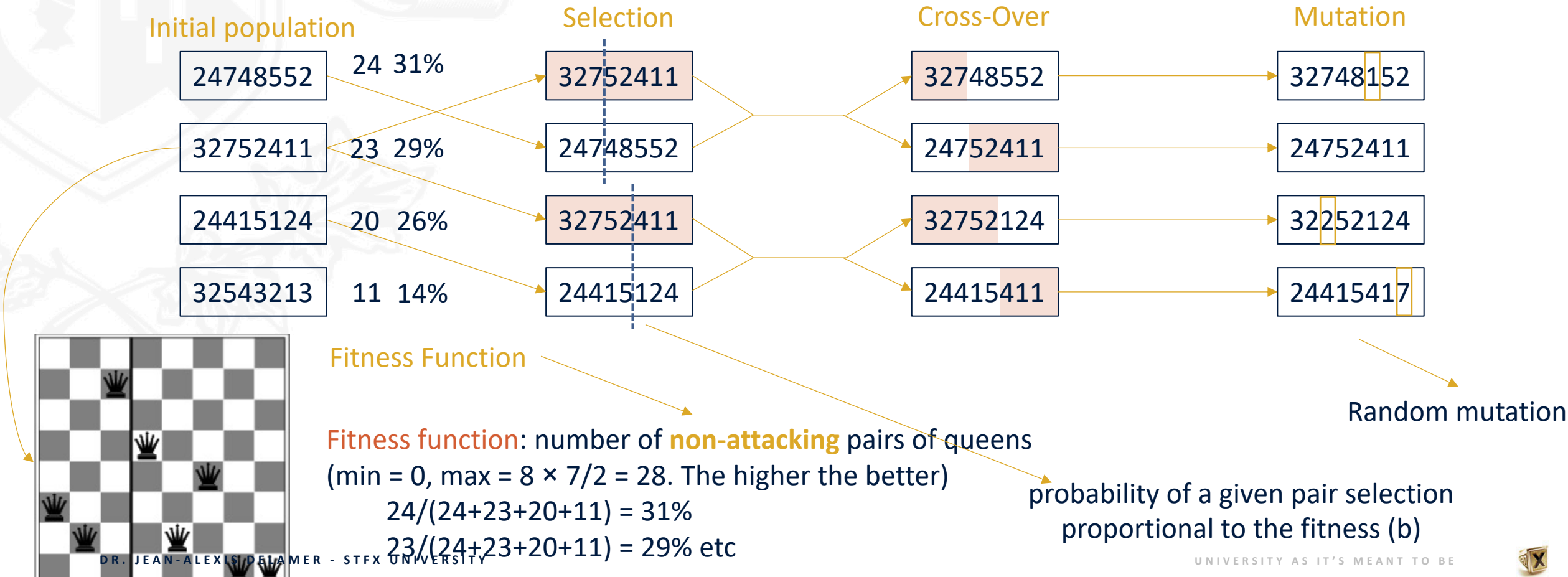
Genetic Algorithms

- Then we iterate.
  - In one generation, the total population fitness changed from 34 to 37.
  - An improvement of 9%.

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

Genetic Algorithms

- Example n-queens.



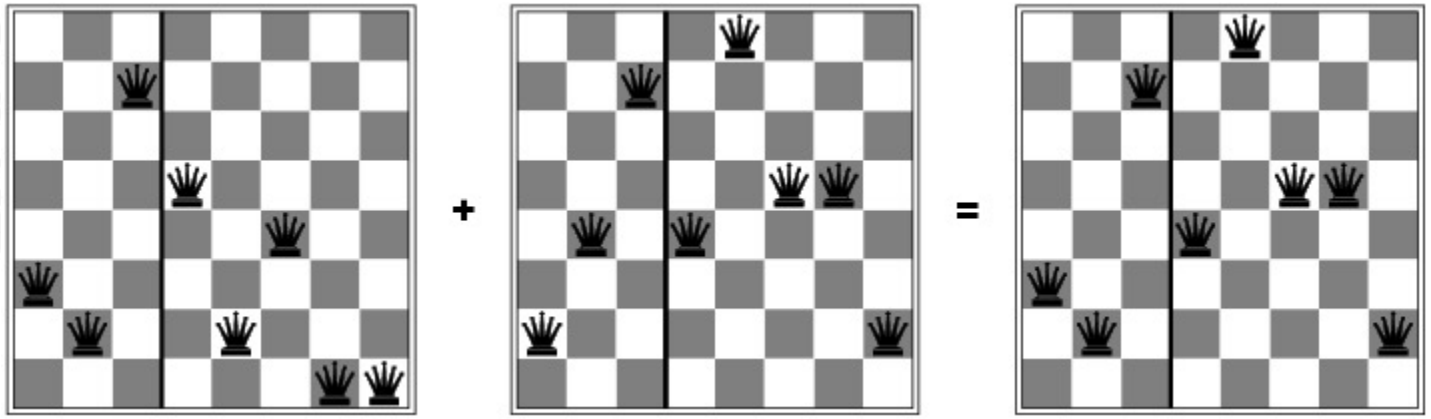
Fitness function: number of **non-attacking** pairs of queens
(min = 0, max = 8 × 7/2 = 28. The higher the better)

24/(24+23+20+11) = 31%

23/(24+23+20+11) = 29% etc

probability of a given pair selection proportional to the fitness (b)

Genetic Algorithms



Genetic Algorithms

- Common representation methods:
  - Binary strings.
  - Arrays of integers (usually bound).
    - Why?
  - Arrays of letters.
  - Etc.

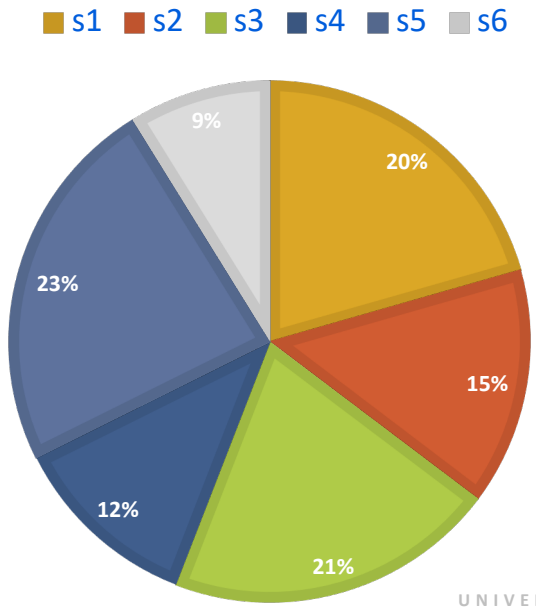
Genetic Algorithms

- 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.

Genetic Algorithms

- 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.

Done in the examples.



Genetic Algorithms

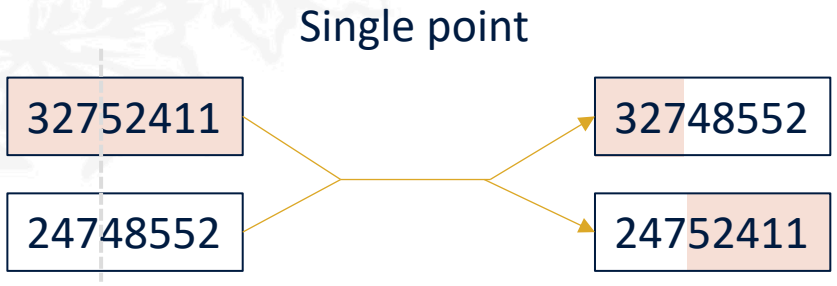
- Elitist selection:
  - Chose only the most fit members of each generation.
- Cut-off selection:
  - Select only those that are above a certain cut-off for the target function.

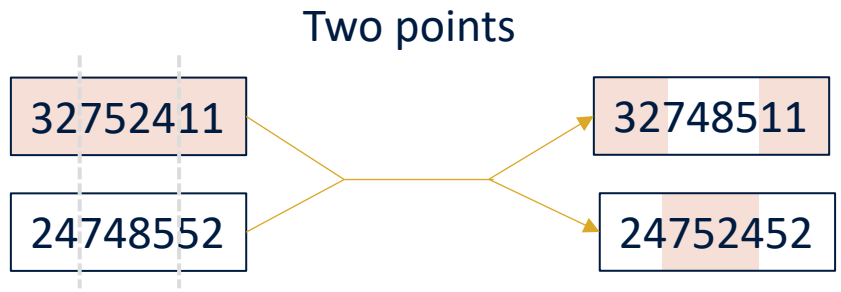
Genetic Algorithms

- Methods of reproduction:
  - There are primary methods:
    - Crossover
    - Mutation
  - Crossover:
    - Two parents produce two offspring
    - Two options:
      1. The chromosomes of the two parents are copied to the next generation
      2. The two parents are randomly recombined (crossed-over) to form new offsprings.
    - In the example we used option 2.

Genetic Algorithms

- Other crossover strategies:
  - Randomly select a single point for a crossover
  - Multi point crossover
  - Uniform crossover

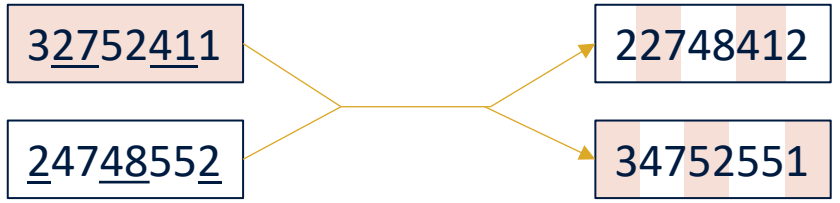




Genetic Algorithms

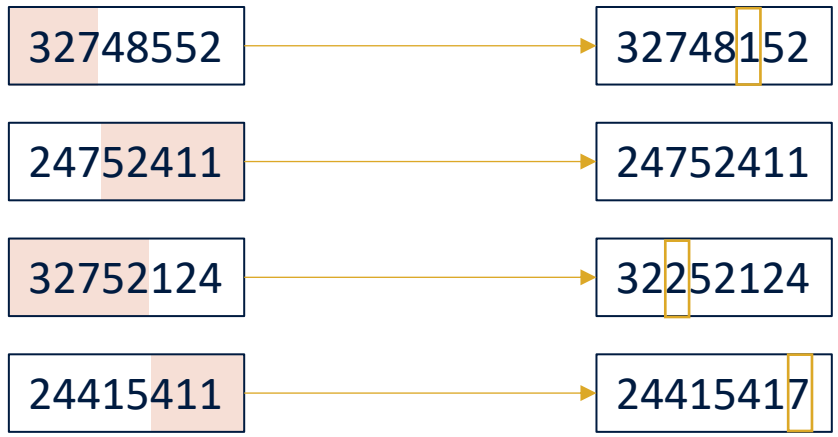
- Uniform crossover:
  - A random subset is chosen
  - The subset is taken from parent 1 and the other bits from parent 2.

Subset: BAABBAAB generated randomly



Genetic Algorithms

- Mutations:
  - Generating new offspring from single parent.
  - Append with a small probability.



Genetic Algorithms

- Exercise:
  - Propose a Genetic algorithm for the Traveling Salesman Problem.
    - State representation.
    - Crossover method.
    - Mutation method.

Genetic Algorithm

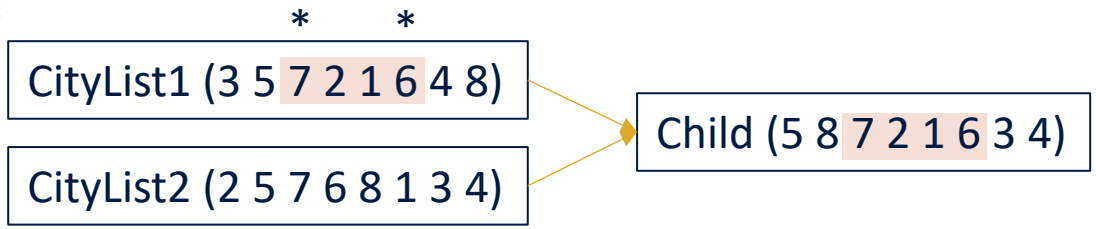
- Representation:
  - An ordered list of city numbers.
  - Known as an order-based GA.

- Examples:
  - 1) London, 2) Venice, 3) Moscow, 4) Singapore, 5) Beijing, 6) Phoenix, 7) Tokyo, 8) Ottawa

  - You can generate a population:
    - CityList1 (3 5 7 2 1 6 4 8)
    - CityList2 (2 5 7 6 8 1 3 4)

Genetic Algorithm

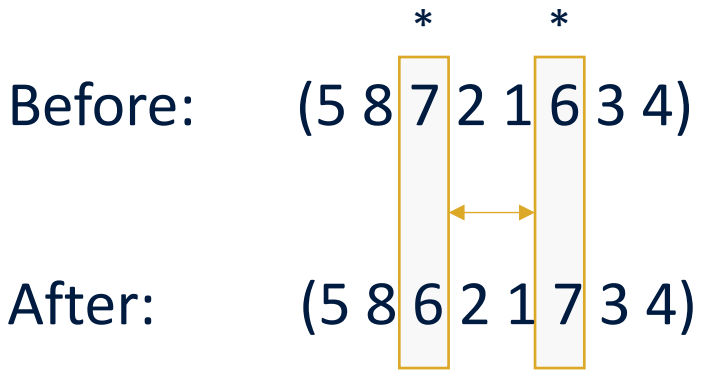
- Crossover:
  - We can use inversion and recombination.



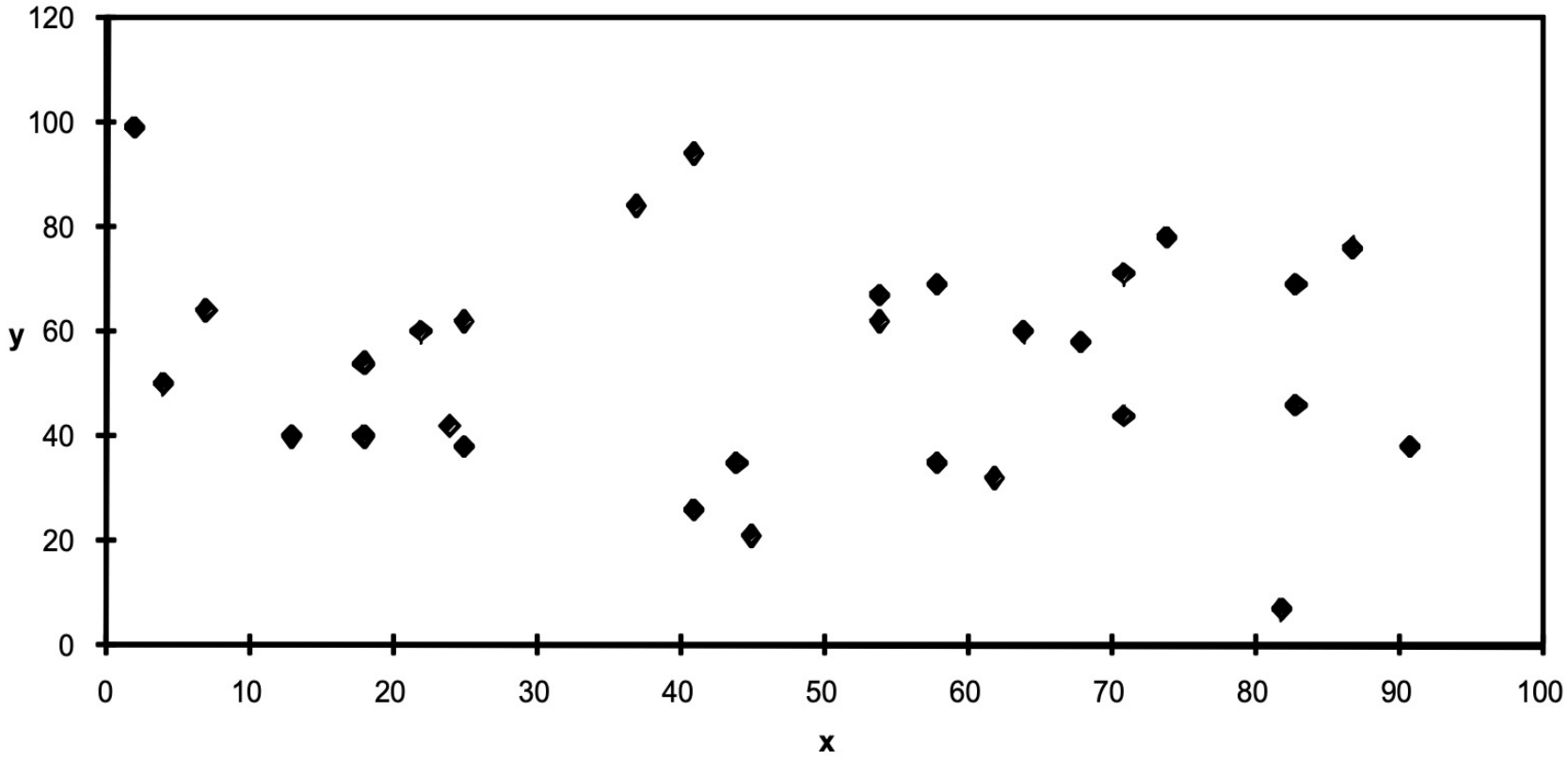
This operator is called the Order1 crossover

Genetic Algorithms

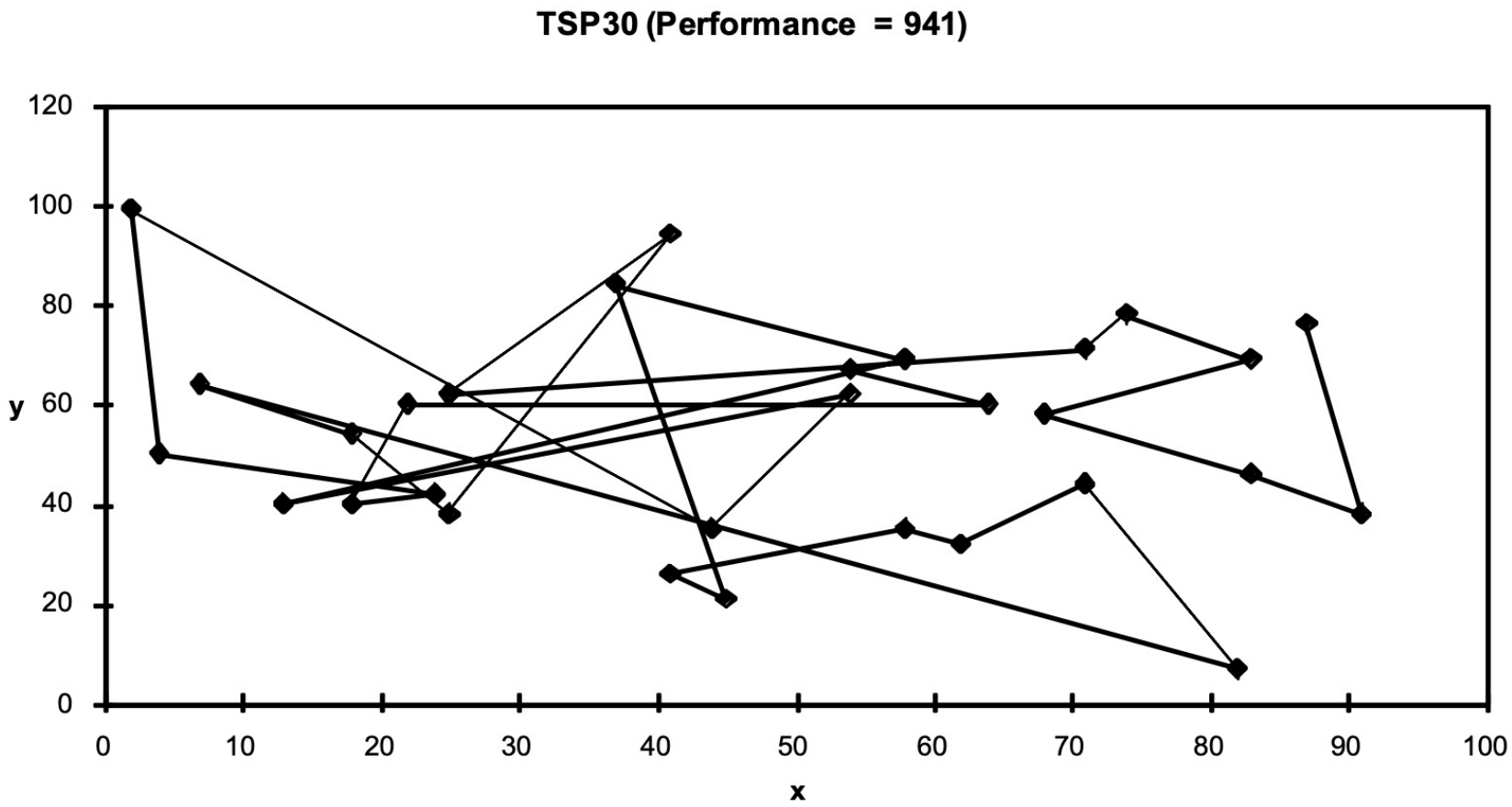
- Mutation:
  - We partially reorder the list.



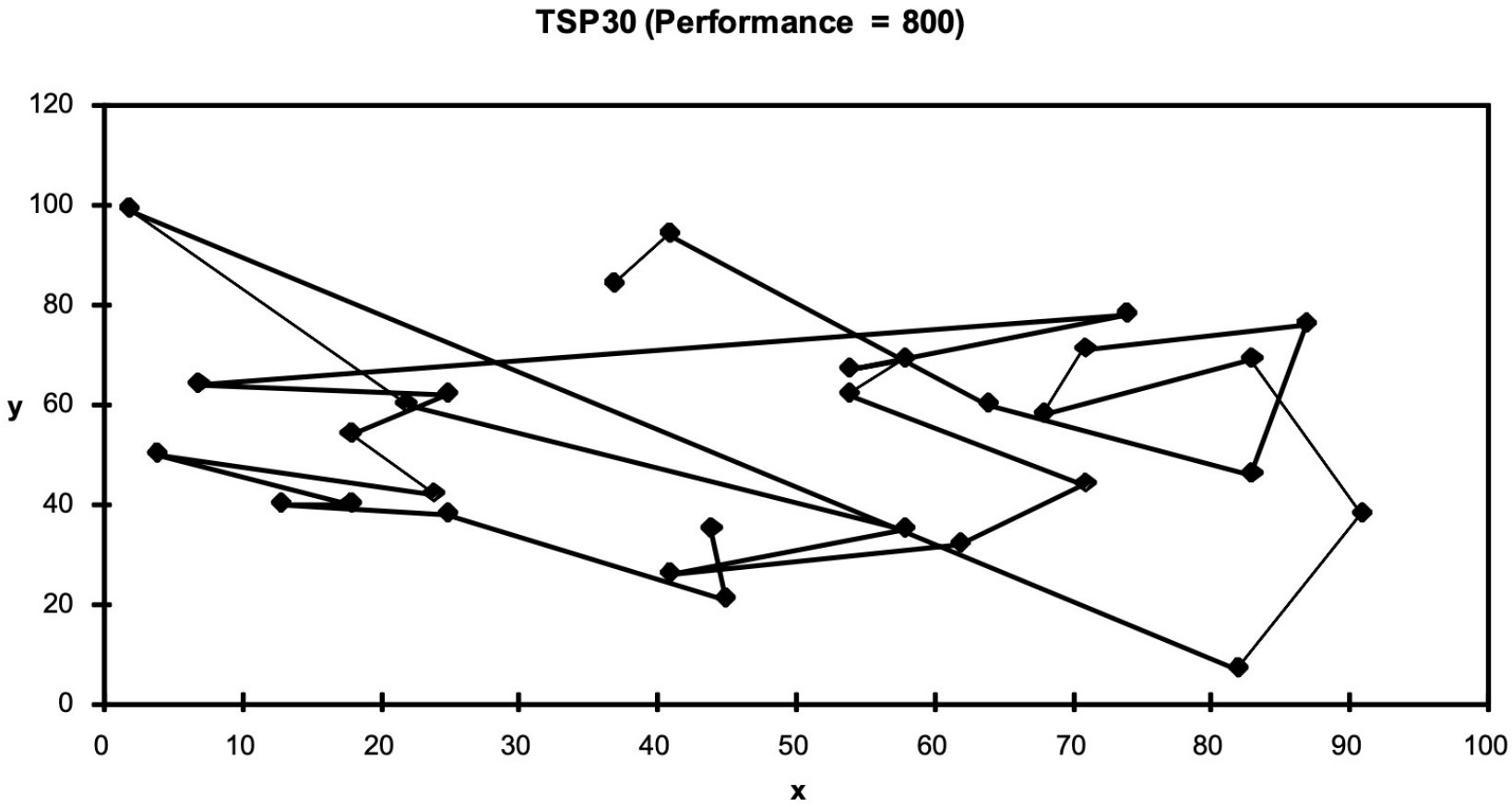
TSP Example: 30 Cities



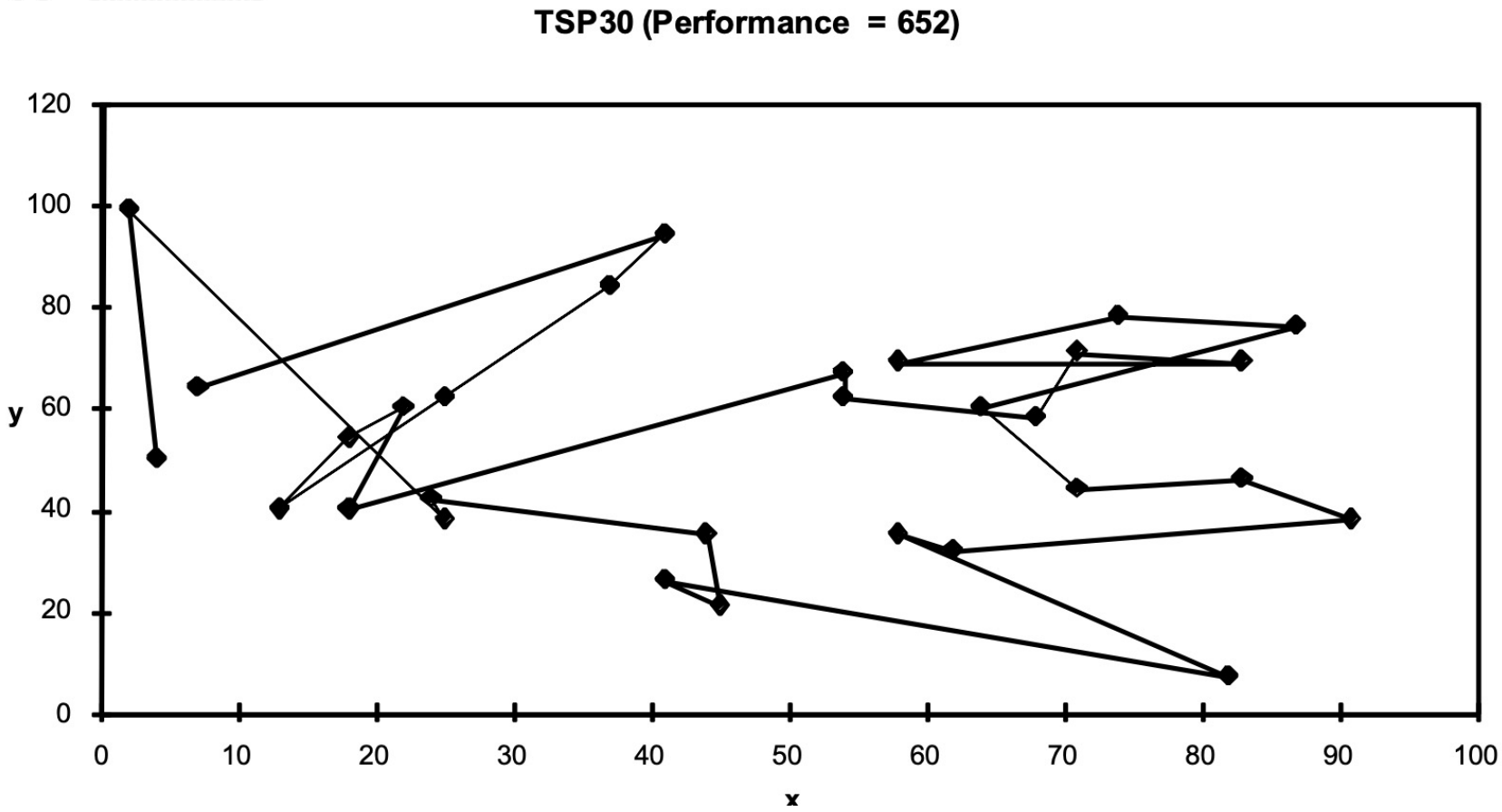
TSP Example: 30 Cities



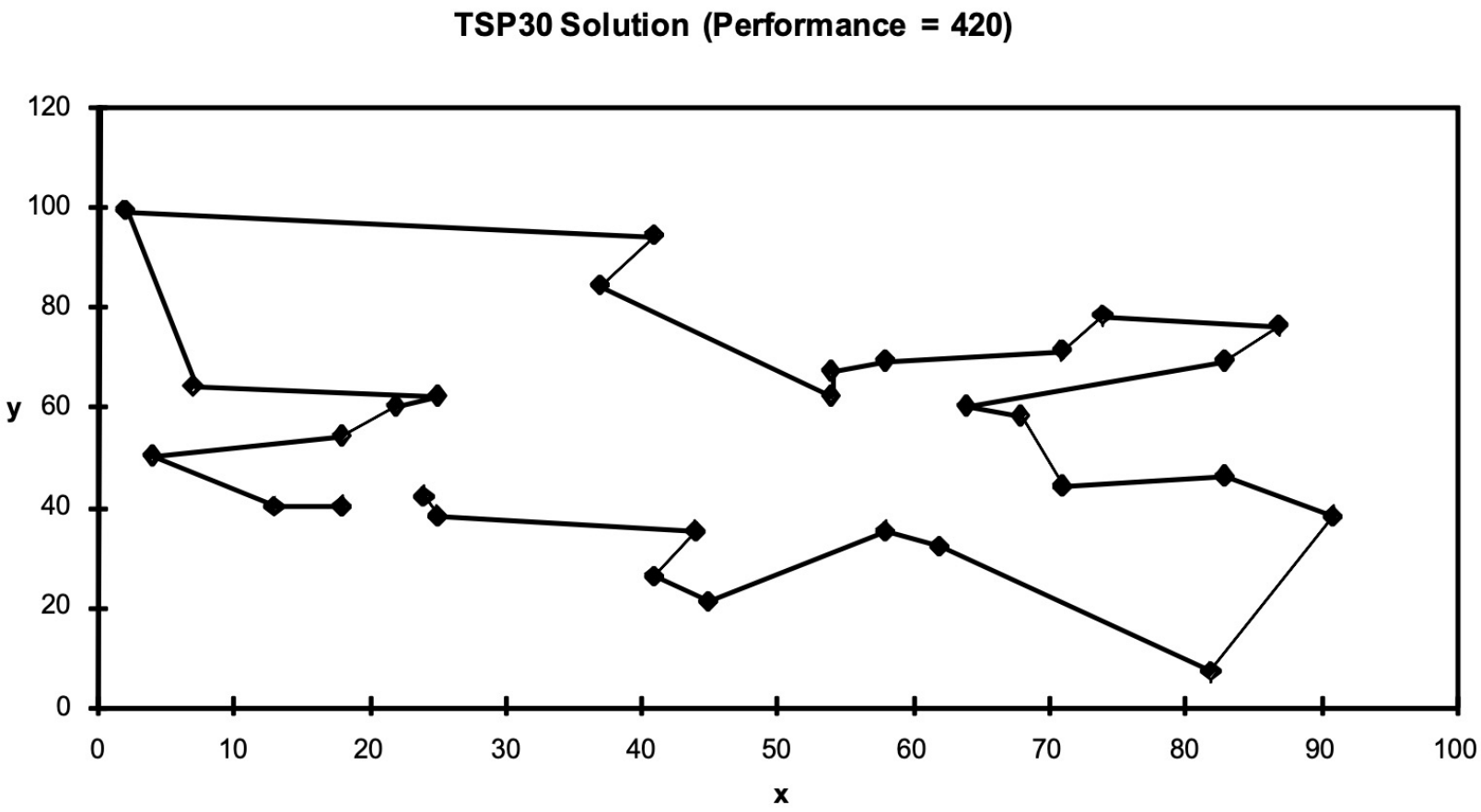
TSP Example: 30 Cities



TSP Example: 30 Cities



TSP Example: 30 Cities



Genetic Algorithms

- Concept is easy to understand.
- Always an answer.
- Flexible building blocks for hybrid applications

- As often with heuristic search, it requires expert knowledge to design a good genetic algorithm.
  - State representation.
  - Crossover method.
  - Mutation method.