Traveling Salesman Problem

#EX14: LP with subtours

1) Try to solve the TSP with a LP matching model. Use the scipy.linprog package. For the cities coordinates use random points.
2) Plot the resulting network with the matplotlib library.
3) Understand the outputs. What happend to the optimal path?
4) What are the alternatives to this formulation and their disadvantages?

$$\min \sum_{ij \in A} c_{ij} x_{ij}$$

$$\sum_{j} x_{ij} = 1$$

$$\sum_{i} x_{ij} = 1$$

$$x_{ij} \geq 0 \quad \forall ij \in A$$

Traveling Salesman Problem

#EX14: Genetic Algorithm

1) Solve the TSP with an original implementation of the Genetic Algorithm (GA).

Simplified Pseudocode:

```
X, Y = GenerateCitiesCoordinates(nº of cities)
initial_population = RandomPopulation(population size)
fitness_results = RankRoutes(initial_population)

while no_improvement below limit:
        parents = PerformSelection(fitness_results, elite_size)
        population = GetNextGeneration(parents, mutation_rate)
        fitness_results = RankRoutes(population)

        if max(fitness_results) > best_fitness:
                save route that corresponds to max(fitness_results)
        else: increase no_improvement
```

Traveling Salesman Problem

#EX14: Genetic Algorithm

Route: tour of cities depending on when will they be visited. For the GA, **chromosome** = route, **gene** = city

1	3	4	2	5	6	7	8

Population: set of different routes to select from. This is the genome.

Always remember the TSP Rules!!:

- Each city should be visited only 1 time.
- After the tour is completed we have to reach the city from which we started from.

Traveling Salesman Problem

#EX14: Genetic Algorithm

Elite size: number of routes that will make it unchanged to the next generation.

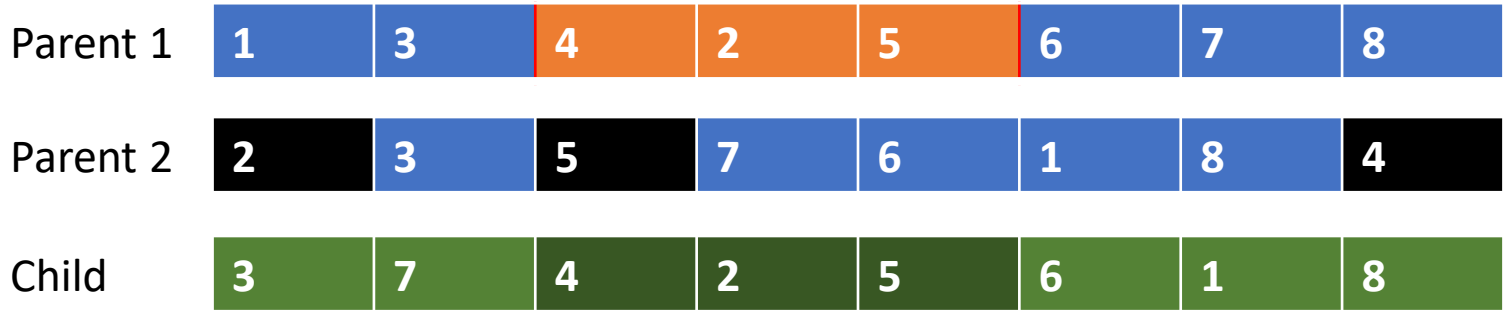
Selection: select parents from a population to get children routes. **We use 'Fitness proportionate selection'.** Chromosomes with higher score have more chances to be selected (analog to a spinning wheel)

Traveling Salesman Problem

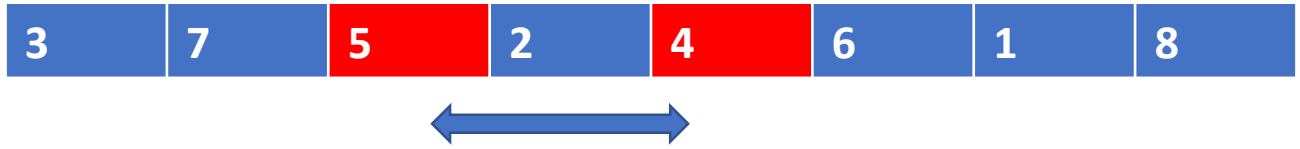
#EX14: Genetic Algorithm

Next generation techniques:

1) Crossover: get a children route from two parents. **We use 'ordered crossover'** in order to preserve the TSP rules.



2) Mutation: with a low rate, make a change in certain routes. **We use 'random swap'.**



Traveling Salesman Problem

#EX15: Greedy Randomize Adaptive Search Procecure

1) Solve the TSP with an original implementation of the Greedy Randomize Adaptive Search Procecure (GRASP).

Simplified Pseudocode:

```
best_route = ConstructRandomSolution()

while no_improvement below limit:
        candidate = GreedyRandomizedConstruction(alpha)
        candidate = LocalSearch(candidate)

        if cost(candidate) < cost(best_route):
                best_route = candidate
```

Traveling Salesman Problem

#EX15: Greedy Randomize Adaptive Search Procecure

1) Solve the TSP with an original implementation of the Greedy Randomize Adaptive Search Procecure (GRASP).

Simplified Pseudocode:

```
best_route = ConstructRandomSolution()

while no_improvement below limit:
        candidate = GreedyRandomizedConstruction(alpha)
        candidate = LocalSearch(candidate)

        if cost(candidate) < cost(best_route):
                best_route = candidate
```

Traveling Salesman Problem

#EX15: Greedy Randomize Adaptive Search Procecure

Greedy Randomized Procedure:

- Step-wise construction of a candidate solution. Construct and select route from restricted candidate list (RCL)

Pseudocode:

```
candidate_route = [initial_city]

while length(candidate_route) < length(city_list):
        calculate features costs (cost of adding feature_i to solution)

        for feat_i in all features costs:
                if (feat_cost_i <= feat_cost_min + alpha * (feate_cost_max – feat_cost_min)
                        add feature_i to RCL

        add random feature from RCL to candidate_route
```

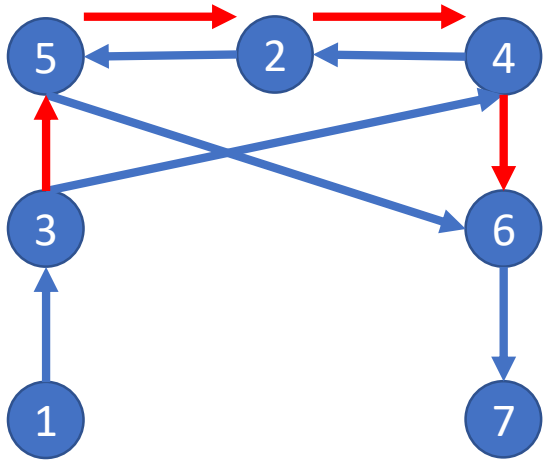
Traveling Salesman Problem

#EX15: Greedy Randomize Adaptive Search Procecure

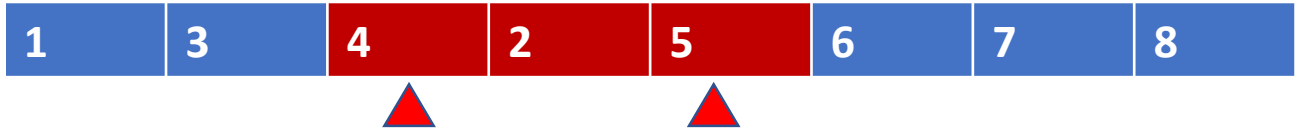
Local Search procedure:

- Any heuristic could be selected to fulfill this procedure. **We will use the stochastic 2-opt heuristic.**

Given any route:



Select two random splitting points:



The tour comprised between them will be swapped: