Exercise 1 - Self-Organizing Systems

Francesco De Nunzio
Roberto Carminati
Pascal Poremba

Time-constrained “Traveling Christmas market visitor”

Christmas is around the corner, and with that comes an optimisation problem! We imagine a traveler in Vienna and he wants to visit all the Christmas markets in as little time as possible. The task is to heuristically optimize a route that minimizes the time used to visit all the markets.
Each market has a different opening and closing time, which are constraints on whether a market can be visited. In addition, at each market the traveler stays a fixed number of minutes (30 minutes). Every market must be visited only once.
As it is impossible to visit all markets in one day, the goal is to optimize and visit as many markets as possible every day, until all the markets will be visited.

The goal is to heuristically optimize a route that maximizes the number of visited markets without calculating all the possible time required by all the combinations. This way would for sure find the best optimal solution but would take too long since we should calculate 32! possibilities.

Data Preparation

As we were only interested in the time distance between each location, we decided to use a matrix of shape 32x32, for the 32 Christmas markets. In this matrix we placed the time distance between each pair of Christmas.
As we at first had trouble with getting a Google Maps API Key (getting our card accepted), so we started by using the Wiener Linien Routes API [https://www.data.gv.at/katalog/dataset/stadt-wien_wienerlinienroutingservice#resources] to iterate through all locations pairwise and retrieve the time of the quickest route by public transport. With this API, it was pretty troublesome to get all times, because we had to use Stop-FInder-Requests to get the nearest public transport and then put the times needed together right. It also had trouble with scenarios where two locations were close enough that walking was the fastest way to go. In this case we were not able to distinguish the two different cases.

Fortunately, our card got accepted by Google later on, so we switched to GoogleAPI [https://maps.googleapis.com/maps/api/distancematrix] which worked way better in our opinion. It was also relatively easy to work with coordinates and we got the time distance matrix without missing values.
In addition, still using GoogleAPI, we also obtained the walking time distance matrix. In this step we had just 4 missing values, which we imputed manually.

Then we computed the optimal matrix, keeping for each combination of Christmas Market the fastest way (public transport or walking) and we keep track of the choice saving it in a matrix (0 if walking, 1 if use public transport).

We decided to first calculate the whole distance matrix in order not to call the API in each step of the algorithm, which would take much more time and resources.

Approach 1: Greedy Algorithm

The greedy algorithm is a simple and intuitive approach to problem-solving that makes locally optimal choices at each stage with the hope of finding a global optimum. In essence, it operates by selecting the best available option at each step, without reconsidering previous choices. The key characteristic of the greedy algorithm is its myopic decision-making, focusing solely on the immediate benefit without anticipating future consequences.

In the specific case of Time-constrained "Traveling Christmas market visitor", this algorithm required some modifications, to respect the time constrained markets: the traveler will always visit the closest and open market, meaning it will have to be open at his arrival, and for 30 more minutes to visit it. The visitor may need more than one day to visit all the markets, but every day is optimized independently creating its own "best" path.

The starting point in the Greedy Algorithm is crucial: the idea we implemented is to always start at the location that opens the earliest not to lose time.
The best path we obtained with the idea above required 2 days until 17:25; in the following output, it is possible to see the order in which the markets should be visited, with the corresponding time of arrival at each one. The time between two markets is always 30 minutes to visit the market, plus the time to move to the next one.

```
Day 1
Visit 1: Location Adventmarkt Mariahilf at 09:00
Visit 2: Location Mittelalterlicher Adventmarkt at 10:08
Visit 3: Location Schönbrunn at 10:51
Visit 4: Location Rathaus at 11:27
Visit 5: Location Stephansplatz at 12:04
Visit 6: Location Altwiener Christkindlmarkt at 12:41
Visit 7: Location Türkenschanzpark at 13:18
Visit 8: Location Christkindlmarkt Floridsdorf (am Franz-Jonas-Platz) at 13:54
Visit 9: Location Adventmarkt Favoriten at 14:31
Visit 10: Location Adventmarkt im Schloss Neugebäude at 15:08
Visit 11: Location Karlsplatz at 15:43
Visit 12: Location Belvedere at 16:24
Visit 13: Location Biobauern-Adventmarkt at 17:03
Visit 14: Location Spittelberg at 17:38
Visit 15: Location Ottakringer Weihnachtszauber at 18:13
Visit 16: Location MQ at 18:50
Visit 17: Location Advent in der Stallburg at 19:26
Visit 18: Location Adventmark Verkehrsmuseum Remise at 20:08
Visit 19: Location Weihnacht im Wald at 20:43
Visit 20: Location Prater / Riesenrad at 21:32
Visit 21: Location Altes AKH at 22:32
Day 2
Visit 1: Location Michaelerplatz at 10:00
Visit 2: Location Adventmarkt Meidling at 10:36
Visit 3: Location Blumengärten Hirschstätten at 11:17
Visit 4: Location Schloss Wilhelminenberg at 12:01
Visit 5: Location Am Hof at 12:39
Visit 6: Location Maria-Theresien-Platz at 13:20
```

```
Visit 7: Location Oper at 14:00
Visit 8: Location IKEA Westbahnhof at14:39
Visit 9: Location Messe at 15:51
Visit 10: Location Palais Liechtenstein at 16:49
Visit 11: Location Winterzauber im Böhmischen Prater at 17:25
```

The time required by this algorithm to run on MacBook Pro M1 is much less than 1 second.

To compare the solution with different starting points, we decided to run the algorithm with every possible starting point (of the first day); as we can see from below, the result above is actually the best one.

```
To visit all the markets: 2 days until 21:28 starting from MQs market
To visit all the markets: 2 days until 19:12 starting from Altes AKHs market
To visit all the markets: 2 days until 19:29 starting from Am Hofs market
To visit all the markets: 2 days until 19:26 starting from Spittelbergs market
To visit all the markets: 2 days until 20:07 starting from Stephansplatzs market
To visit all the markets: 2 days until 20:33 starting from Opers market
To visit all the markets: 3 days until 11:30 starting from Türkenschanzparks market
To visit all the markets: 2 days until 18:48 starting from Maria-Theresien-Platzs market
To visit all the markets: 2 days until 20:04 starting from Blumengärten Hirschstättens
market
To visit all the markets: 2 days until 19:21 starting from Palais Liechtensteins market
To visit all the markets: 2 days until 17:39 starting from Belvederes market
To visit all the markets: 2 days until 20:04 starting from Karlsplatzs market
To visit all the markets: 2 days until 21:15 starting from Messes market
To visit all the markets: 2 days until 18:26 starting from Altwiener Christkindlmarkts
market
To visit all the markets: 2 days until 19:43 starting from Biobauern-Adventmarkts market
To visit all the markets: 2 days until 18:25 starting from Rathauss market
To visit all the markets: 2 days until 18:33 starting from Schönbrunns market
To visit all the markets: 2 days until 20:10 starting from Prater / Riesenrads market
To visit all the markets: 2 days until 18:35 starting from Mittelalterlicher Adventmarkts
market
To visit all the markets: 2 days until 19:21 starting from Michaelerplatzs market
To visit all the markets: 2 days until 20:45 starting from Schloss Wilhelminenbergs market
To visit all the markets: 2 days until 20:25 starting from Advent in der Stallburgs market
To visit all the markets: 2 days until 18:28 starting from Adventmarkt Mariahilfs market
To visit all the markets: 3 days until 11:30 starting from IKEA Westbahnhofs market
To visit all the markets: 2 days until 20:05 starting from Christkindlmarkt Floridsdorf (am
Franz-Jonas-Platz)s market
To visit all the markets: 3 days until 11:30 starting from Adventmark Verkehrsmuseum
Remises market
To visit all the markets: 3 days until 12:24 starting from Weihnacht im Walds market
To visit all the markets: 2 days until 18:50 starting from Adventmarkt Favoritens market
To visit all the markets: 2 days until 19:47 starting from Winterzauber im Böhmischen
Praters market
To visit all the markets: 2 days until 19:00 starting from Adventmarkt Meidlings market
To visit all the markets: 2 days until 20:11 starting from Ottakringer Weihnachtszaubers
market
To visit all the markets: 2 days until 20:36 starting from Adventmarkt im Schloss
Neugebäudes market
```

As we can see, the approach to always start the day visiting the market that opens the earliest worked, because it had the best duration compared to all the other starting points. To run all the possible starting points, the algorithm still took less than 1 sec.

While the greedy algorithm is known for its simplicity and efficiency, it does not guarantee the most optimal solution globally. Usually, making choices solely based on immediate gain led to suboptimal outcomes. However, this algorithm was easy to implement, fast to run, and gave us an idea of what a good solution looks like.

Approach 2: Ant Colony

The Ant Colony Optimization (ACO) algorithm is a nature-inspired optimization technique that draws inspiration from the foraging behavior of real ant colonies. Developed based on the observation that ants can find the shortest path between their nest and a food source, ACO has been applied to solve various combinatorial optimization problems.

In ACO, artificial ants iteratively construct solutions by probabilistically selecting paths based on the amount of pheromone deposited on them. Pheromones serve as a form of communication, with higher concentrations indicating more favorable paths.
Usually, shorter paths accumulate more pheromone due to positive feedback, but in our scenario the positive feedback is given to “longer” paths, meaning that most markets have been visited.

To prevent premature convergence to suboptimal solutions, ACO incorporates a pheromone evaporation process, simulating the natural decay of pheromones in the environment. This allows the algorithm to explore a broader solution space. Periodically, a global update adjusts pheromone levels on all paths based on the quality of the solutions found, intensifying exploration in promising regions.

We implemented a Time Constrained ACO, where the ants must constantly check if they have enough time to visit the next location; if they don’t, they will visit the remaining markets the following day, until all the locations will be visited.

The complexity of this algorithm comes from the many parameters:

- Number of ants that will traverse the problem space: the more ants, the more the exploration.
- Number of iterations: the more the iterations, the more it converges.
- Alpha, higher value gives more importance to the pheromone information.
- Beta, higher value gives more importance to the distance information.
- Evaporation rate is the rate at which the existing pheromones evaporate.
- Q, the higher, the more the pheromones are left.

Tuning all these parameters increases the complexity of the algorithm. In our scenario, since we didn’t already have any detailed knowledge about the algorithm, we looped and tried all the possible combinations of parameters (729 combinations) from the sets of parameters below, looking for the best combination.

```
different_ant_numbers = [10,50,100]
different_iteration_numbers = [10, 50, 100]
different_alpha_values = [1,3,5]
different_beta_values = [1,3,5]
different_evaporation_rates = [0.3, 0.5, 0.8]
different_Q_values = [1, 5, 10]
```

We noted that, even fixing the parameters, every run could have quite different results: for this reason, every combination of parameters is run 7 times to take the average time at which the tour ends. The time required by this process is 290 mins on MacBook Pro M1. The best parameters created a tour which end time average is at 15:55 :

- ants: 100
- iterations: 100
- alpha: 3
- beta: 5
- evaporation: 0.3
- Q: 10

Other valid parameters are:

- n_ants=100, n_iterations=50, alfa=3, beta=5, evaporation=0.8, Q=1 → 2 days until 16:11
- n_ants=10, n_iterations=100, alfa=3, beta=5, evaporation=0.8, Q=10 → 2 days until 16:20
- n_ants=50, n_iterations=10, alfa=3, beta=5, evaporation=0.8, Q=5 → 2 days until 16:23

As we can see, even if some parameter combinations are similar in some values, they are very different in the others, making it hard to find a trend.

It is important to highlight that running the algorithm 100 times, even specifying the best parameters, the results were very different between them. The tour average end time shifted from 15.55 (value calculated before with 7 iterations) to 17:11 (calculated as mentioned with 100 iterations); the variance is 6:33 hours. For this reason we believe that presenting the best parameters requires many more runs of the algorithm, but anyway the result would have a very high variance.

In our case the goal is to present the best route, so we took track of the best solution during the tuning of the parameters, where the algorithm ran 5103 times (729 combinations * 7 iterations) keeping changing the parameters.

The best path was found with 10 ants, 50 iterations, alpha = 3, beta = 5, evaporation = 0.3, and Q = 5, and it ended at 14:57 (+30 minutes to visit the last market). As we can see the parameters are very different from “the best ones”, but the algorithm randomly found the best solution.

```
----> Day 1
Christkindlmarkt Floridsdorf (am Franz-Jonas-Platz) at 09:00
Adventmarkt Favoriten at 09:37
Adventmarkt im Schloss Neugebäude at 10:14
Winterzauber im Böhmischen Prater at 10:50
Maria-Theresien-Platz at 11:25
Karlsplatz at 12:03
Stephansplatz at 12:42
Prater / Riesenrad at 13:19
Michaelerplatz at 13:54
Adventmarkt Meidling at 14:30
Oper at 15:07
Adventmark Verkehrsmuseum Remise at 15:42
Weihnacht im Wald at 16:17
MQ at 16:52
Advent in der Stallburg at 17:28
Schönbrunn at 18:03
Rathaus at 18:39
Am Hof at 19:19
Ottakringer Weihnachtszauber at 19:55
```

```
Türkenschanzpark at 20:34
Messe at 21:10
Altes AKH at 22:08
----> Day 2
Mittelalterlicher Adventmarkt at 09:00
Adventmarkt Mariahilf at 09:41
Spittelberg at 10:22
Altwiener Christkindlmarkt at 11:00
Blumengärten Hirschstätten at 11:40
Belvedere at 12:15
Biobauern-Adventmarkt at 12:54
Palais Liechtenstein at 13:31
Schloss Wilhelminenberg at 14:20
IKEA Westbahnhof at 14:57
```

Conclusion

In conclusion, it is relevant to highlight that most of the ACO solutions (even with random parameters) are better than the best ones found with the Greedy algorithm.
In addition, from our experience it is probably not worth it to tune all the parameters to find the best combination, since the solutions always have a very high variance. It would make more sense to just run the algorithm encouraging exploration, in order to “randomly” find a good solution.