Self-Organizing Systems: Exercise 1

Florian Mende, 12017067
Jan Widerhofer-Kaukal, 11907081

November 2025

1 Introduction

This report describes the approach, solutions, and findings of applying **genetic algorithms** and **ant colony optimization** techniques to the **time-constrained "Travelling Christmas market visitor"** problem.

2 Approach

2.1 Data gathering and preparation

We started with the list of Christmas markets provided by the lecture team. As a first step, minor manual clean-up steps were taken:

- The URL provided for *MQ* was not a valid Google Maps link. Since the provided URL suggested, that the intended market is "Weihnachtsquartier", we replace the URL with a link to *Urania*, since the market was relocated there.
- The URL provided for *Belvedere* was not a valid Google Maps link. We updated the URL to match the URL of Belvedere.
- The URL of *Messe* has incorrect coordinates in Google Maps. As we only realized that after all of our experiments were done, we kept the original coordinates here.

We then created a unified dataset that stores Name, Google Maps URL as well as opening times of each market. Using the Google Maps API, we obtained travel times between all possible pairwise combinations of markets for both walking and via public transit. We also extract coordinates from the API that were later used for vizualization purposes.

This results in an asymmetric distance matrix for the given list of markets, which was used as an input for our algorithms.

2.2 Ant Colony Optimization (ACO)

The ACO algorithm was implemented using the Python library spades, which uses an XMPP to asynchronously manage multiple agents. Our implementation uses three types of agents: ants, a pheromone manager, and a coordinator.

The optimization problem is split up into several days. On the first iteration, all markets might be included in the constructed routes. On the second day, all markets that have been visted on the first day are removed and a new ant colony is spawned from scratch.

2.2.1 AntAgent

Each ant asynchronously constructs a feasible route by sampling the next market with probability proportional to pheromone intensity (alpha) and an inverse-distance heuristic (beta). Before committing to a hop, the agent checks time-window feasibility by projecting arrival and service completion against each market's opening hours, to make sure that only markets that can be fully serviced are considered. Ants get the pheromone levels via messages to the pheromone manager.

2.2.2 PheromoneManager

The pheromone manager maintains a full matrix indexed by market IDs. After every iteration the manager applies uniform evaporation, aggregates the tours submitted in that round, and reinforces only the best-so-far path (elitist update).

2.2.3 CoordinatorAgent

A dedicated coordinate agent orchestrates the iterations: it broadcasts a `start_iteration` signal to all ants, waits until each has reported completion, instructs the pheromone manager to apply evaporation/deposits, and finally queries the current best solution before moving on. This synchronization prevents stale tours from polluting the pheromone matrix and ensures every iteration has a clear begin/end boundary.

2.3 Genetic Algorithms

We implemented a genetic algorithm using the Python library deap.

In our approach, each candidate itinerary is represented as a permutation of the available markets. The algorithm evolves this population to maximize the number of markets that can be feasibly visited within a single day.

Each individual (itinerary) is decoded into an ordered tour and evaluated through a lightweight scheduling simulation that advances time along the route. Markets that cannot be fully serviced within their time windows are excluded from the fitness score, allowing infeasible candidates to be naturally filtered out during evaluation.

The evolutionary process follows the standard DEAP workflow. We begin with an initial population sampled uniformly across the space of all possible permutations, ensuring broad coverage of potential routes.

Genetic operators are applied as follows:

- Crossover: exchanges contiguous subsequences between parents while preserving valid permutations.
- Mutation: randomly shuffles selected indices to introduce diversity without breaking the permutation constraint.
- Selection: tournament selection promotes high-performing itineraries while maintaining competition across the population to prevent premature convergence.

Throughout the configured number of generations, the algorithm records fitness statistics — including the mean, variance, and best values — to monitor progress.

A hall-of-fame archive keeps track of the best itinerary found so far. After convergence, the best chromosome is transformed back into an executable daily plan by re-running the feasibility simulation. This produces both the final route and the corresponding visit count.

For multi-day planning, the solver is executed iteratively. After each day, markets that have already been visited are removed from the dataset, and the optimization is rerun on the remaining ones.

3 Results

To evaluate the performance and find good hyperparameter combinations for the algorithms, we decided to implement grid search. The parameter ranges can be seen in Table 1. Since we only optimize the route on a per day basis, we ran grid search for with a time window of one day only and then proceeded with the optimal parameters with a time window of 3 days.

3.1 Grid Search

For ACO we observed that the best runs by score and runtime used 20 ants for 5 iterations with the highest heuristic weight β in range. This suggests that the heuristic is good enough for this problem that even the lowest number of iterations in range is sufficient to find the best solution. Visiting 18 markets in 1 day is potentially even the optimum, since both ACO and GA converged to it. Comparing the runtime of ACO to GA we observed a stark difference as the average time for one run in the ACO algorithm takes roughly 1 minute while it is about 10 seconds for GA. However, this doesn't necessarily reflect the efficiency of either of the two algorithms, since *spade* and our implementation in particular introduces overhead through communicating via XMPP while

deap is a presumably optimized implementation for genetic algorithms. The results for the 1 day ACO grid search can be seen in Table 2.
In Table 3 the 1 day grid search results for of the genetic algorithm can be seen. There seems to be a correlation between N_{tourn} the tournament size, and execution time suggesting that this parameter is the deciding factor for how long one run takes. Unlike in ACO, the genetic algorithm runs that find the (potentially optimal) solution that visits 18 markets require the highest tournament size in range, while the fastest runs trivially use the lowest one. Notably, the best runs all use an individual mutation probability P_{ind} of 0.1 while the other parameters vary, suggesting that P_{ind} of 0.1 is both a good value choice and a deciding factor.

Table 1: **Hyperparameter Grid Search Space.** The table shows the parameter ranges for both the Genetic Algorithm (GA) and Ant Colony Optimization (ACO).

Algorithm	**Parameter (Symbol)**	**Values Tested**
Genetic Algorithm (GA)		
	Population Size (N_{pop})	{300, 500, 800}
	Generations (N_{gen})	{300, 500}
	Crossover Probability (P_{cx})	{0.6, 0.7, 0.8}
	Mutation Probability (P_{mut})	{0.1, 0.2, 0.3}
	Indiv. Mutation Prob. (P_{ind})	{0.05, 0.1, 0.2}
	Tournament Size (N_{tourn})	{3, 5}
	Total Combinations	*324*
Ant Colony Optimization (ACO)		
	Number of Ants (N_{ants})	{20, 40}
	Iterations (N_{iter})	{5, 10, 15}
	Initial Pheromone (τ_0)	{1.0}
	Pheromone Decay (ρ)	{0.3, 0.5, 0.7}
	Pheromone Weight (α)	{1.0, 1.5}
	Heuristic Weight (β)	{2.0, 3.0, 4.0}
	Reward Multiplier (R)	{2.0, 5.0}
	Total Combinations	*216*

Table 2: **Ant Colony Optimization Results. (1 Day)** Global statistics and comparison of highest-scoring vs. fastest-converging parameter sets. Note: For all top runs shown, $N_{ants} = 20$, $N_{iter} = 5$, and $\tau_{init} = 1.0$.

Global Statistics (N=216)			
Metric	**Value**	**Metric**	**Value**
Success Rate	100.00%	Mean Time/Run	62.94 s
Mean Score	17.02	Total Duration	13 532.48 s
Max Score	18.00		

Top Configurations Analysis						
	Parameters				**Performance**	
Rank	α	β	Decay (ρ)	Reward	**Score**	**Time (s)**
Top 3 by Score (Highest Quality)						
1	1.0	4.0	0.5	5.0	**18.00**	31.08
2	1.5	4.0	0.5	2.0	**18.00**	31.17
3	1.5	4.0	0.5	5.0	**18.00**	31.08
Top 3 by Time (Fastest Execution)						
1	1.0	2.0	0.5	5.0	16.00	**31.06**
2	1.5	4.0	0.3	5.0	16.00	**31.06**
3	1.0	3.0	0.5	5.0	16.00	**31.06**

Table 3: **Genetic Algorithm Optimization Results. (1 Day)** Global statistics and comparison of highest-scoring vs. fastest-converging parameter sets.

Global Statistics (N=324)			
Metric	**Value**	**Metric**	**Value**
Success Rate	100.00%	Mean Time/Run	10.12 s
Mean Score	16.23	Total Duration	3268.51 s
Max Score	18.00		

Top Configurations Analysis								
	Parameters						**Performance**	
Rank	P_{cx}	P_{mut}	P_{ind}	N_{pop}	N_{gen}	N_{tourn}	**Score**	**Time (s)**
Top 3 by Score (Highest Quality)								
1	0.6	0.3	0.1	300	500	5	**18.00**	6.79
2	0.8	0.3	0.1	300	500	5	**18.00**	7.52
3	0.6	0.2	0.1	500	300	5	**18.00**	6.92
Top 3 by Time (Fastest Execution)								
1	0.6	0.1	0.1	300	300	3	16.00	**3.77**
2	0.6	0.2	0.05	300	300	3	17.00	**3.79**
3	0.6	0.1	0.2	300	300	3	17.00	**3.79**

Note: P_{cx} = Crossover Prob., P_{mut} = Mutation Prob., P_{ind} = Indiv. Mutation Prob., N_{pop} = Population Size, N_{gen} = Generations, N_{tourn} = Tournament Size.

3.2 Multi-Day Routes

Using the best parameters found from grid search, which can be seen in Section 3.1, we ran the optimization algorithms with 3 days time. We picked 3 days since it is possible to visit all markets in this time frame and none of our runs fond a solution that completes in 2 days. As can be seen in Figure 1 the ACO algorithm found three routes that intuitively can be split into an inner-city, west and east routes. To us these tours seemed quite close to what a human would choose. Interestingly the routes the genetic algorithm found in Figure 2 seem more chaotic although they are valid routes and visit all markets. It should be noted that this particular run did not find the best route visiting 18 markets that it found in grid search because of the algorithm inherent randomness.

Figure 1: Ant Colony Optimization 3 Day Routes

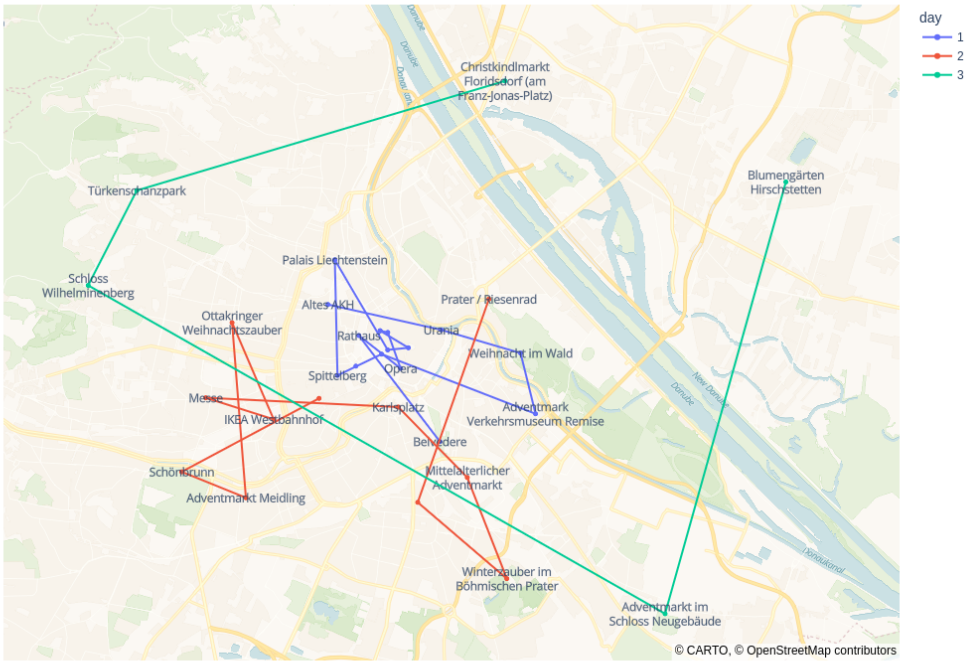

Figure 2: Genetic Algorithm 3 Day Routes