Self-Organizing Systems

Assignment 1: Exercise GA/Ants/Agents

Time-constrained “Travelling Christmas market visitor”

Group 13
Benicio Daniel – 12038369
Philipp Schott – 12132552

November 2025

Contents

1 Generel Overview

This project applies an Ant Colony Optimization (ACO) approach to compute efficient, time-constrained routes through Vienna's Christmas markets. Multiple colonies explore different starting points, with each ant constructing routes based on heuristic DNA, pheromone learning, and real Google Maps travel times. Over many generations, mutation, selection, and a shared pheromone pool improve the solutions, and visual analyses track the algorithm's progress. The final result is the best route that visits as many markets as possible within opening hours and daily limits.

We tackled the "Time-constrained Travelling Christmas Market Visitor" problem by splitting responsibilities: initially one of us focused on ant behaviour and the other on collecting travel data from Google Maps; later, both contributed to testing and integrating the full pipeline in 'main.py'.

Travel times were retrieved via the Google Maps API and wrapped in a custom service class that provides ants with travel durations and reachable next locations. The Ant Optimizer spawns colonies, which in turn spawn ants that perform local path search using both genetic DNA inheritance and pheromone-based reinforcement.

2 Getting google maps data

For getting the google maps data we had a few different considerations. First we used the the Google Maps API as instructed, with the distance matrix method of said API. Iterating over the provided list in the assignment, it resulted in 992 edges with walking duration and distance. Some names were altere for the google maps API to be more consistend, as the original list names were often not found by the api.

As a second step we reduced the edge amount. We estimated that if the time to walk from location A -> C times a margin (in our case 10 %) was greater or equal to A -> B -> C we could delete the edge A -> C. This means if there is a christmas market on the way between two other markets that does only increase the time walking, we assumed that we could delete the original longer path. Another condition is that every node had to retain at least the two shortest edges, as to avoid dead ends for ants and avoid illogical graph construction. This was the case with spittelberg, as it is so close to mariahilf, maria-theresianplatz and MQ, that it often ended up with only one edge to MQ in these algorithms. This step resulted in a reduced number of edges to 170, which reduced the problemspace by a huge amount.

In the third step, we estimate that if a public transport trip was half the walk time, we replace the transit time with public transport estimates via the API. These steps were taken in this order to avoid other complications with public transport times and then the edge reduction, which resulted in weird and illogical graphs. After some experimentation to get more reliable results we used distance from location to location for the edge reduction. The resulting graph is shown in Appendix Figure 1.

Ants and Related Classes

The implementation of the ant-based components followed a bottom–up approach: core behaviour was developed first, and higher-level structures were built on top of it. Accordingly, the following subsections describe each class in the order in which they were implemented.

2.1 ant.py

Each ant is initialized with a set of arguments and configuration parameters. Besides fixed attributes such as its name, an ant receives access to the maps service object, a predefined starting market, a start time, a stay duration for each market, and a default time limit (set to 11 pm, corresponding to the latest closing time in the dataset). Initially, each ant also starts with DNA = None and generation = 0, indicating that no inherited path information is present yet. To enable flexible local search behaviour, different mutation modes (e.g., mode 1 for random choices) were implemented and are described later in this report. A verbose flag was added to optionally trace an ant's decisions for debugging purposes. All initialization parameters are passed from the Ant Optimizer down to the colonies and finally to each individual ant (as explained in the following sections).

After initialization, each ant cycles through two core behaviours during training: it first evaluates all feasible next steps and then performs a move based on the available options. In the method `evaluate_possibilities`, the ant queries the maps service object to obtain all reachable neighbouring markets from its current position, including the corresponding travel times, pheromone levels, and opening and closing times of each destination. The ant then filters these options by checking whether a market has already been visited and whether it would arrive after opening and still have enough time to complete its stay duration before the market closes.

All valid options identified in the evaluation step are passed to the ant's `move` method, where different decision behaviours are implemented. Each ant selects its next market according to one of four mutation modes, which are determined by the `mutation` value assigned to all ants within the same cohort (as initialized by the Ant Optimizer). An ant may choose randomly (mutation $= 1$, primarily used for testing), or select a destination based on DNA, pheromones, or a combination of both. To accomplish this, each option is assigned a weight that increases for destinations favoured by the current mode (e.g., higher pheromone levels or DNA matches). Once a choice is made, the ant updates its position, time, and visited markets, and repeats this process until no further moves are possible or the time limit is reached. The completion of this procedure is later referred to as one generation of the ant.

Once the multi-day logic is activated after a given generation threshold, each ant is allowed to use more than one day (max_days > 1). Whenever an ant has no valid moves left on the current day but still has remaining days, it starts a new day at a randomly chosen Christmas market that it has not visited before, with its time reset to that market's opening time. On each new day beyond the first, the ant automatically switches to mutation mode 3 (purely pheromone-based movement), reflecting that inherited DNA is no longer reliable once the ant is dropped into an unfamiliar region of the graph.

2.1.1 DNA

After all ants in a colony have completed their route for one generation, a one-point crossover is applied to the paths of pairs of ants within the same colony. The resulting combined subsequences form the DNA that is assigned to the ants of the next generation. During route construction, the DNA is incorporated into the decision process: when evaluating possible moves, destinations that also appear in the inherited DNA sequence receive an increased weight in the selection function. This bias increases the likelihood that an ant will reproduce beneficial segments of its parents' trajectory (e.g., if a parent moved from MQ to Spittelberg, its offspring is more likely to perform the same transition), thereby guiding exploration based on previously successful routes.

2.1.2 Pheromones

As explained previously, we constructed a proprietary Google Maps class containing all 170 reduced edges, including their walking or transit times and, additionally, the pheromone values associated with each edge. This Google Maps object is created at the top level and passed down to all ants, ensuring that any edge can be "pheromonized" by all ants during the optimization process.

The motivation behind this design is the following: once the multi-day logic is activated, ants may start a new day at a completely different Christmas market that they have not previously visited. This drastically enlarges the solution space, making it difficult for purely genetic information to guide the ant reliably, as the neighbourhood the ant is dropped into may have never been explored by its ancestors. For the first day, we rely mainly on genetics (and optionally pheromones), but once the ant moves to an unfamiliar region of the graph, the pheromones deposited by the ants that previously explored that region help guide it toward promising routes.

At the end of each generation in every colony (see below), the colony calls the pheromone update method of the Google Maps class. For each ant, we pass the list of all edges (i, j) that the ant traversed along with its fitness value. After applying the global evaporation (multiplying all pheromone values by the decay factor), we add new pheromone proportional to the ant's fitness and inversely proportional to the travel time of the edge:

$$\tau_{ij} \leftarrow \text{decay factor} \cdot \tau_{ij} + \frac{\text{pheromone constant} \cdot \text{fitness of ant}}{\text{duration of walking}(i,j)}$$

During the ant's decision process, the pheromone level of an edge is multiplied by the inverse of its travel time. The result is then normalized by dividing by the sum over all available options. Pheromones are raised to an exponent α, while travel time is raised to an exponent β, allowing either information source to be weighted more strongly. These are standard ACO hyperparameters, which we did not tune in this exercise but could be varied in future experiments.

$$P(\text{choose } j) = \frac{(\text{pheromone}(i,j))^{\alpha} \left(\frac{1}{\text{travel time}(i,j)}\right)^{\beta}}{\sum_{l \in \mathcal{J}(i)} (\text{pheromone}(i,l))^{\alpha} \left(\frac{1}{\text{travel time}(i,l)}\right)^{\beta}}$$

2.1.3 Multiple Days Logic

After a configurable generation threshold, the algorithm enables multi-day behaviour for all ants. From this point onward, each ant is allowed to use more than one day (max_days > 1) to continue exploring the search space. If an ant reaches a point where no valid moves remain for the current day (because of conditions above), it automatically starts a new day: it is relocated to a randomly selected Christmas market that it has not visited yet, and its time is reset to that market's opening time.

Because DNA-based inheritance loses meaning once an ant is dropped into an entirely different region of the graph, the ant therefore switches before that happens to pure pheromone-based behaviour (mutation = 3) for all remaining days. This ensures that exploration on later days is guided solely by globally accumulated pheromone information rather than by local genetic bias.

2.2 ant_colony.py

Each Ant Colony spawns a group of ants that all share the same starting market and starting time. This design keeps the genetic recombination consistent, since all ants in a colony explore the search space from an identical initial condition. The number of ants per colony is configurable, and for convenience each ant receives an automatically generated name based on the starting market and its index.

Within each colony, selection is performed using the fitness function:

$$len(ant.visited) * 100 - ant.current\_min/60.$$

This formulation prioritises ants that visit more markets, while still rewarding those that complete the same number of markets in less time.

If the minimum fitness in the colony is negative, all fitness values are shifted upward so that the smallest becomes 1, ensuring valid roulette-wheel probabilities. Survivors are then selected via roulette-wheel selection (with replacement), as described in the lecture. Based on these survivors, random pairs of ants are chosen and combined via a simple one-point crossover to produce new DNA sequences for the next generation. A validity check ensures that the chosen crossover point produces a feasible transition: the cut is only accepted if the edge (i→j) exists in the reduced Google-Maps graph. If the edge is not valid, the algorithm simply tries the next candidate from the shuffled list of possible crossover points.

After that we let every ant in that colony perform one generation and return the paths to update the pheromones. At this point the path is an array consisting of the markets in the order the ant has terversed them (eg. [m2, m1, m3]). For easier update we pass these paths as edges (eg. [(m2, m1) (m1, m3) corresponding to above example]).

2.3 ant_optimizer.py

The Ant Optimizer is responsible for creating and coordinating multiple Ant Colonies, where each colony consists of ants that all start from the same market and at the corresponding opening time. During initialization, the method initialize_colonies distributes the colonies evenly across all available markets by repeatedly shuffling the list of market indices. This ensures that every market is used as a starting point, even if the number of colonies is larger than the number of markets. Each colony then spawns its ants with identical initial parameters, except for their unique names.

The method run_one_generation iterates through all colonies and lets each colony execute its full movement phase, meaning that all ants explore until no further valid moves are possible. The resulting paths from all colonies are collected and used to update pheromone levels in the shared Google-Maps graph. Since all ants across all colonies contribute to the same pheromone map, the optimizer maintains a global learning signal across the entire population.

After all ants have finished their movement, the optimizer advances the algorithm to the next generation through advance_to_next_generation. This triggers the genetic step inside each colony, where selection and crossover produce the next generation of ants. The optimizer therefore controls the high-level evolution cycle, while each colony manages its own internal reproduction and search process. If multi-day behaviour is activated, the method set_ants_multiple_days propagates the allowed maximum number of days to every colony and to every ant within them.

3 Training Process

The core training logic of our system is implemented in the central testing function test1() in the main.py document, which orchestrates the complete execution of an experiment. The function evaluates the behaviour of the Ant Colony Optimization pipeline across three progressively enriched phases. Each phase modifies the configuration of the Ant Optimizer and repeats the same training procedure under different conditions. This allows us to isolate the effects of pheromones, mutation modes, and multi-day logic. Most of the code in this section generates plots and performance statistics, which are discussed later in the Results chapter.

Below these shared mechanics, the training setup is structured into three configurable phase transitions, each building on the insights gained in the previous one. The first switch, `time_to_cull`, reduces the number of colonies after a chosen generation by keeping only a fixed top percentage of starting markets. This allows the algorithm to focus its resources on the most promising regions of the search space while still maintaining diversity early on. The different tests later simply change the initial mutation behaviour of the ants.

The second transition, `time_to_switch_pheromones`, reflects the idea that DNA-based mutation is most meaningful when ants repeatedly explore similar neighbourhoods. Early in training, inherited path segments can effectively guide the search within the local surroundings of each starting market. However, as the search progresses, global information becomes more valuable. For this reason, the system gradually shifts toward pheromone-driven behaviour, allowing ants to benefit from shared global experience accumulated across all colonies rather than relying on purely local genetic history.

The final transition, `time_to_set_multiple_days`, enables the multi-day logic. Once activated, ants that become stuck due to time or opening-hour constraints may begin a new day at an unvisited market. This drastically enlarges the effective search space, and by activating it only after pheromone guidance is well established, the system ensures that ants navigating unfamiliar regions can rely on globally reinforced pheromone trails rather than local DNA biases that no longer apply. Together, these three staged transitions allow the experiment to move from local learning, to global reinforcement, to full cross-graph exploration in a controlled and interpretable way.

3.1 Training configurations

To evaluate the behaviour of the different ant decision strategies, we conducted three controlled experiments using identical training conditions. For all three runs, we fixed the number of generations to 100, used 40 ants per colony and 32 colonies corresponding to the 32 different starting positions, applied a stay time of 30 minutes, a cut off of 30%, and aligned all phase-transition points:

- Cull colonies after generation 30
- Switch to pheromone-driven behaviour at generation 50
- Enable multi-day logic at generation 80

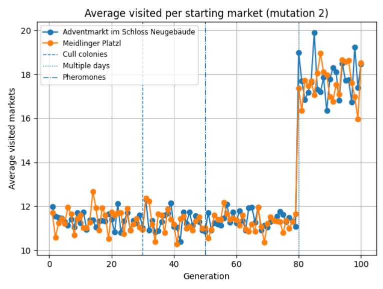
Except for the mutation mode, all parameters were held constant. This allows us to isolate the effect of the three behavioural strategies:

- Mutation 2 — Pure DNA
- Mutation 3 — Pure Pheromones
- Mutation 4 — Hybrid DNA + Pheromones

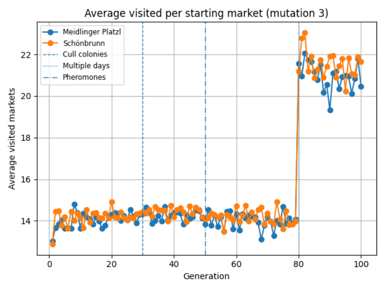
All configurations use the same training pipeline, producing fitness curves, visited-market statistics, and best-path visualisations that differ only by the ants' decision strateg. After identifying the best performing mutation, we ran an extended experiment with 200 generations and 50 ants per colony and a 40% cutoff to test whether performance continues to improve under longer training.

4 Results

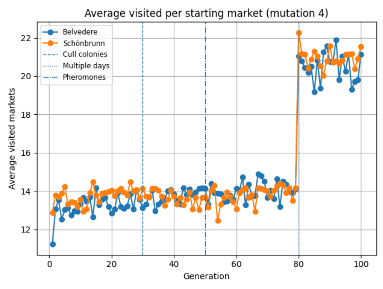
In the following, we compare how the three mutation strategies perform under identical training conditions.All experiments with 100 generations required roughly 10 minutes of runtime each, while the extended 200-generation experiment (with more ants per colony) took approximately 25 minutes, mainly due to the larger number of evaluated routes per generation.



M2: DNA Only



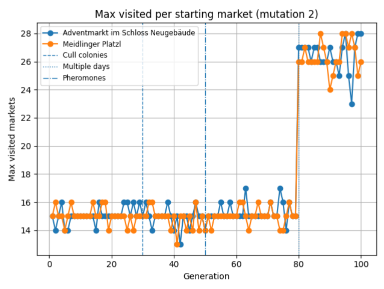
M3: Pheromones Only

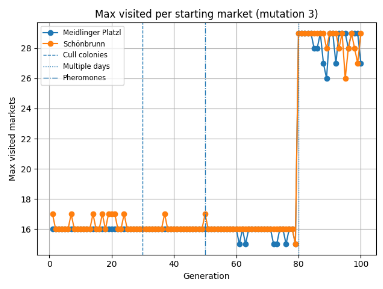


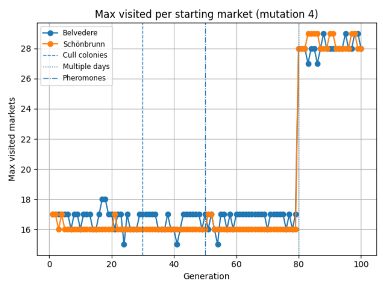
M4: Hybrid

Across the three decision strategies, we see that pheromones-only (M3) produces the highest overall average number of visited markets. The hybrid approach (M4) shows the clearest upward learning curve, steadily improving until the multi-day switch at generation 80, after which it performs almost as well as pure pheromones. In contrast, the DNA-only strategy (M2) remains noticeably below both alternatives and shows only limited improvement over time.

A small but informative detail is that the best-performing starting market depends heavily on the strategy. Both M3 and M4 consistently rank Schönbrunn among the top origins, while M2 selects different markets entirely. This suggests that DNA-based inheritance is less effective at identifying globally advantageous regions of the graph, reinforcing that pheromone-driven learning is the more reliable mechanism in this problem setup.





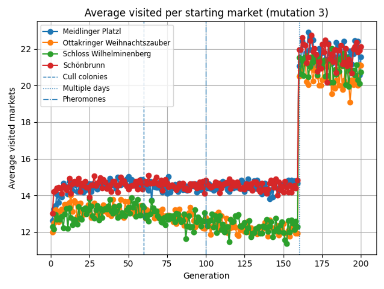


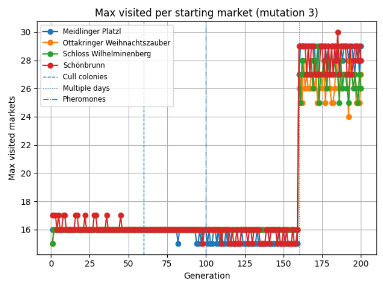
M2: DNA Only **M3: Pheromones Only** **M4: Hybrid**

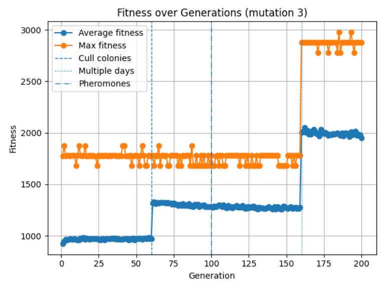
Across all three decision strategies, the maximum-visited-plots reveal a clear performance hierarchy. With pheromones only (M3), ants achieve a remarkably stable plateau of 29 out of 32 markets wihtin a two days, already immediately after the multi-day switch. Since this value barely improves afterward, it suggests either a natural upper bound imposed by opening-hour constraints or that further optimisation would require structural rather than iterative improvements. The hybrid method (M4) consistently reaches 28 markets, occasionally peaking at 29, but shows slightly less stability, likely due to the remaining influence of DNA, which introduces mild variability but no sustained gains. In both strategies, Schönbrunn remains among the strongest starting positions.

By contrast, the DNA-only configuration (M2) performs considerably worse: even after two days it stabilises around 26–27 markets, never reaching the performance range of the pheromone-driven variants. This confirms that purely genetic guidance is insufficient for navigating the globally interconnected graph, and motivates the sequential training setup in which DNA is used only early for local exploration, before being replaced by more reliable pheromone-based exploitation.

It is also possible that the superior performance of pheromone-based strategies partially reflects their cumulative nature: since pheromones are reinforced over many generations, they may naturally grow into a stronger global signal than DNA, which is repeatedly reshuffled through crossover.







In the extended 200-generation experiment (with all phase transitions proportionally delayed), the system shows behaviour broadly consistent with the 100-generation runs, but with a few notable refinements. In the maximum-visited plot, we observe that the algorithm occasionally reaches 30 out of 32 markets, demonstrating that such a near-complete route is theoretically attainable under the 30-minute stay-time constraint. The two best starting markets clearly separate themselves early in training, as seen particularly in the average-visited curves, even though this difference is less visible in the maximum-visited statistics. The fitness curve displays the expected staircase pattern: each major phase transition culling, pheromone emphasis, and multi-day activation produces a measurable jump in overall fitness.

While the long run yields a slight improvement over the 100-generation scenario (largely due to the occasional 30-market route), the overall structure of the results remains similar. Some starting markets gradually drop off as training progresses, suggesting that they are inherently less favourable under the given constraints. Overall, the extended run confirms the stability of mutation 3 while showing only marginal late-stage gains.

5 Conclusion

Across all experiments, pure pheromone behaviour (mutation 3) proved to be the most effective strategy. DNA-based guidance helped only in the earliest exploration phase, but lost usefulness once ants moved into new areas of the graph, where global pheromone information clearly dominated. The hybrid approach performed well but never surpassed pure pheromones, suggesting that local genetic bias is not essential for this problem.

The extended 200-generation run confirmed this pattern: although performance improved slightly (with occasional 30-market routes), the gains did not justify the much longer runtime. Convergence occurs early, and additional training adds little value.

Overall, the algorithm shows that a pheromone-focused ACO with multi-day logic yields the strongest results, and that only a small set of starting markets consistently leads to high-performing routes.

6 Further Improvements

Several extensions and refinements could potentially improve the performance and robustness of our approach:

- **Hyperparameter tuning:** The pheromone–distance exponents (α, β) and the DNA weighting factor γ were left at default values. Systematic tuning or automated search (e.g., grid search or Bayesian optimisation) could significantly change the balance between exploration and exploitation.
- **Improved genetic operators:** Our crossover is limited to a simple one-point cut, which often fails because legal edges do not exist at the crossover boundary. More advanced operators (e.g., edge-preserving crossover, segment recombination, or probabilistic path blending) may make DNA inheritance more meaningful. It remains an open question whether genetic guidance is beneficial for this problem at all.
- **Mixed mutation strategies within one cohort:** Instead of assigning one mutation type to all ants in a colony, different subgroups (e.g., DNA-focused and pheromone-focused ants) could coexist and mutually reinforce each other, providing a richer search dynamics.
- **Bidirectional traversal:** Currently, ants rely on directed edges as generated by the Google Maps API. Allowing explicit evaluation of reverse paths—or symmetrising the graph—could open additional feasible transitions and reduce dead ends.
- **Waiting behaviour:** Ants are forced to skip markets if they would arrive too early. Introducing the ability to wait until a market opens might lead to longer but more complete routes, especially in multi-day scenarios.

Appendix

Additional plots generated during the experiments can be found in the repository under the ‘plots/‘ directory, organised by the corresponding test configurations.

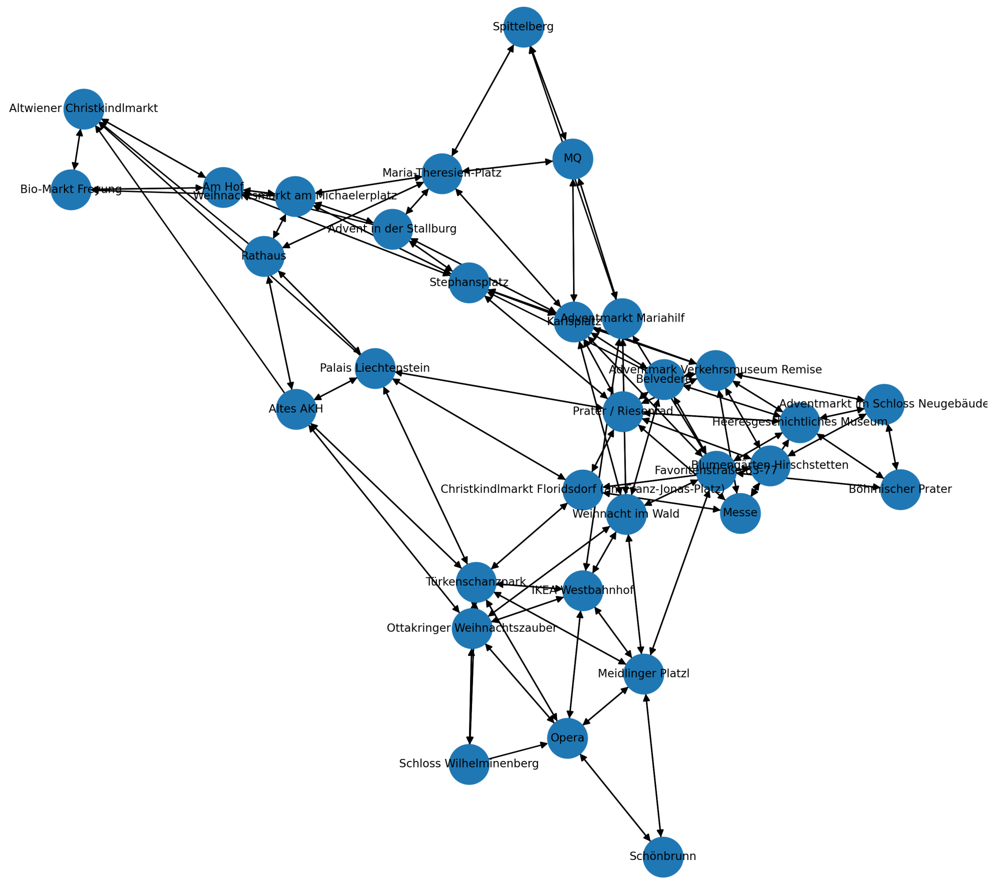


Figure 1: Reduced graph after preprocessing steps.