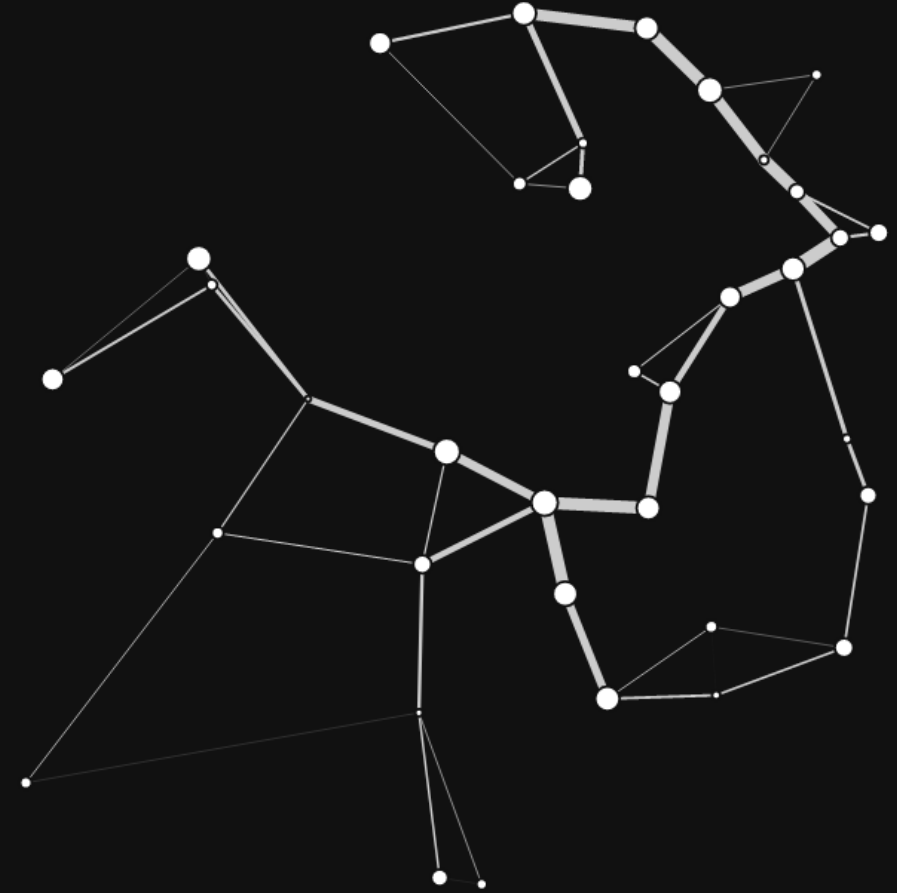
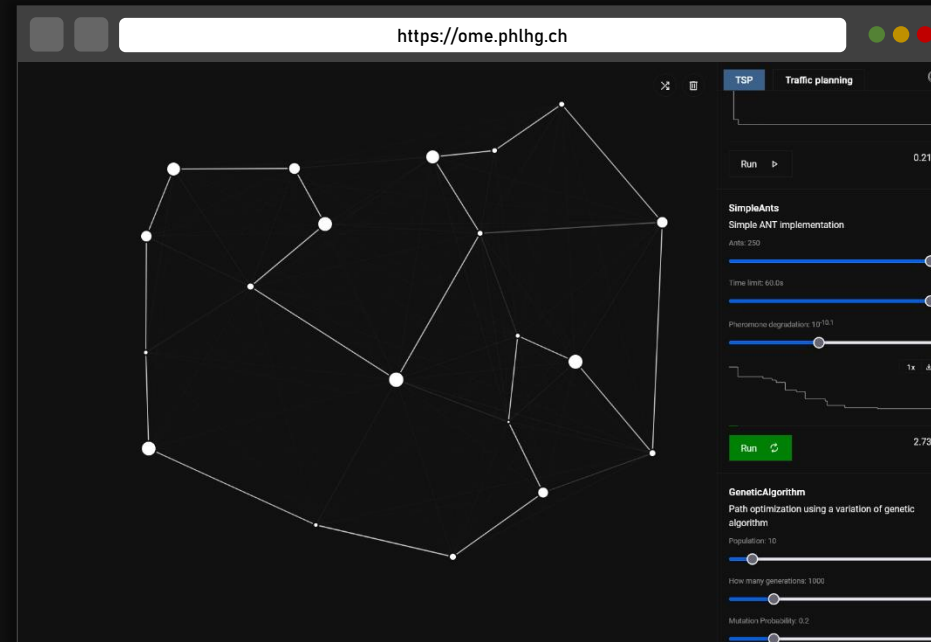


Traffic Problems

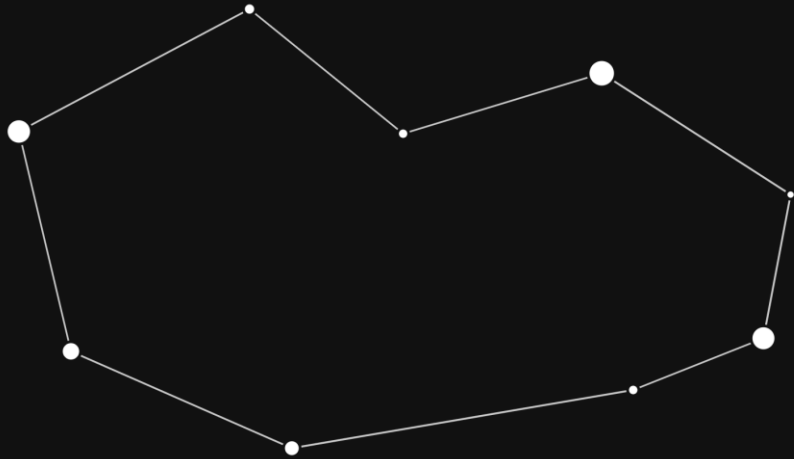
by Aaron Hodel, Philippe Hugo and Philipp Scherer



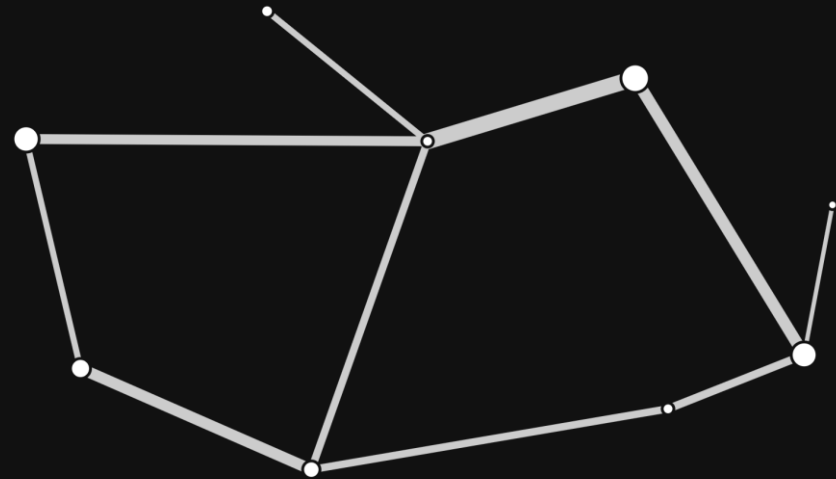


All problems and methods can be
experimented with on:
<https://ome.phlhg.ch>

Problems



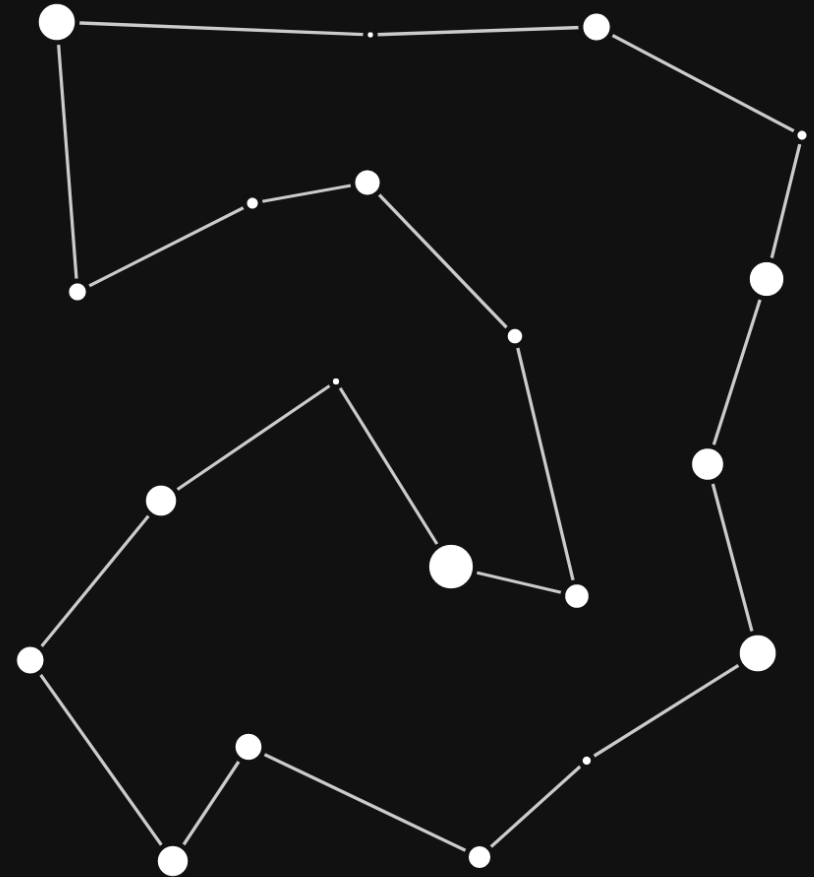
Traveling Salesman Problem



Traffic Problem

Traveling Salesman Problem

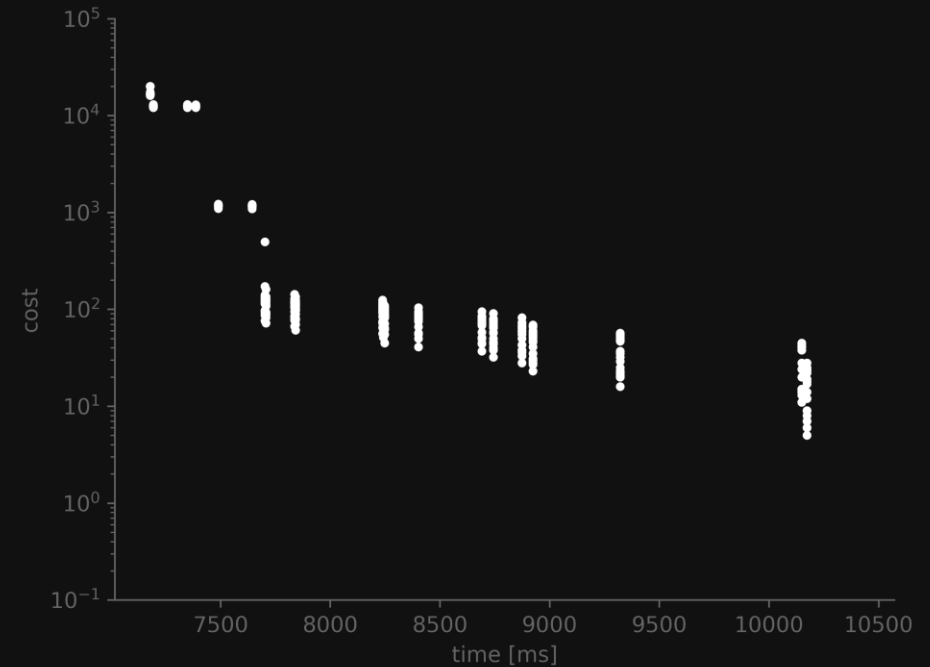
Given nodes $v \in V$, find a permutation of nodes **minimizing the length of the cycle** resulting from connecting consecutive nodes in the permutation. The length of the cycle is calculated by summing up the lengths of the contained edges.



Bruteforce

As the name suggests, the Bruteforce approach calculates the length of the cycle for all $n!$ possible permutations of nodes and takes the best permutation.

But since $n!$ grows fast with the number of nodes, this approach becomes **infeasible even for small number of nodes** (In our implementation this happens around 9 nodes).



Bruteforce is **very slow** in improving its solutions due to its lack of Strategy.

Simulated Annealing

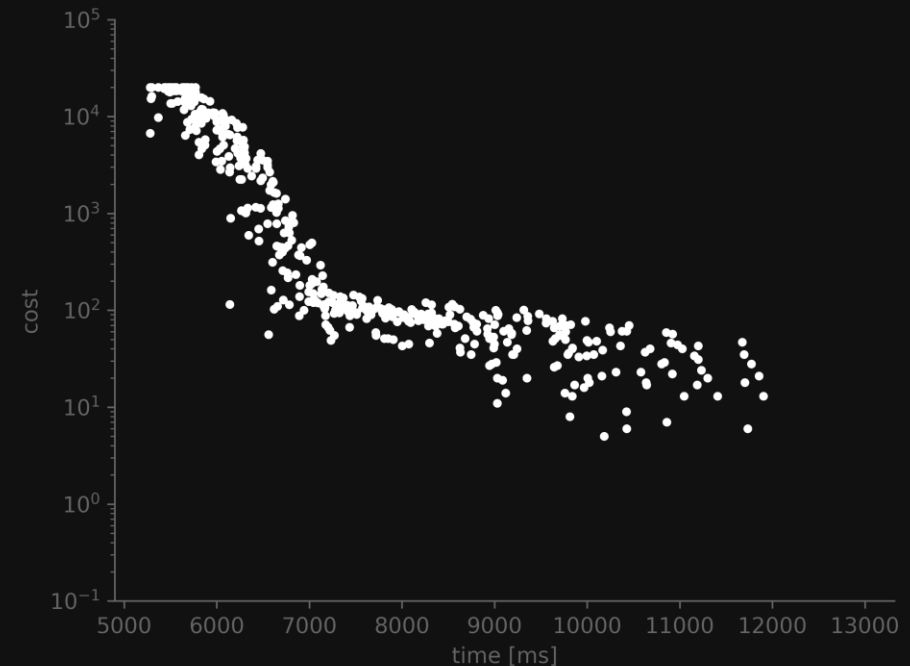
We use time as a mean for temperature to control the degree of exploration.

SA avoids getting stuck in local minima by **exploring more variations of cycles** (even worse ones) while on high temperature in the beginning.

A random step will select a permutation from the neighborhood of the current permutation.

The neighborhood is defined as the permutations reachable by swapping two nodes in the sequence.

Otherwise, we use the same algorithm as in lecture.



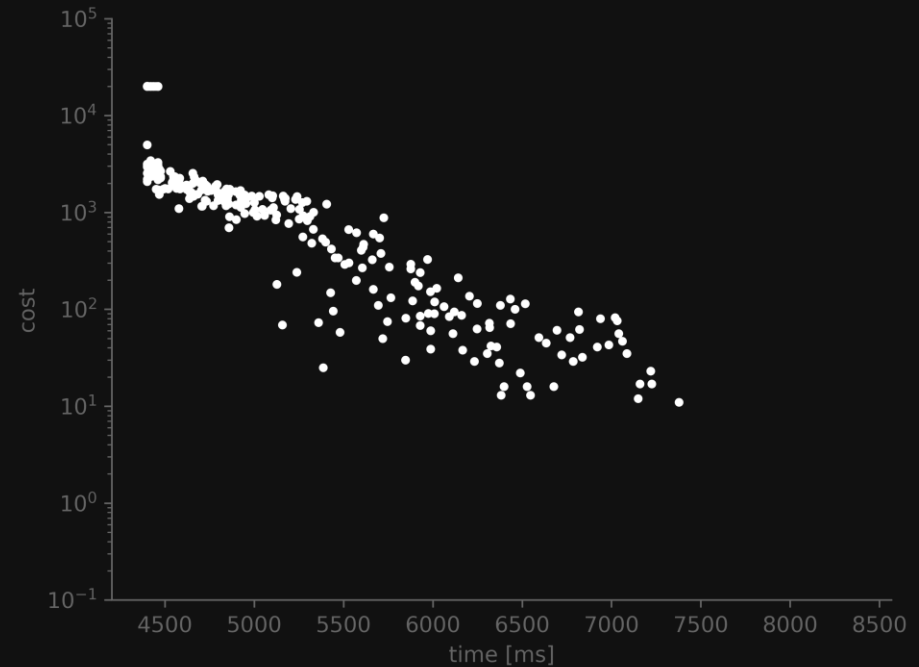
SA initially is able to make **fast improvements**, but as the temperature decreases and the solution gets better the exploration is less favored, which leads to **less improvement**.

Ant Colony

Starting from a random start node each ant iteratively searches for the next best node according to the pheromones and distance of the edges until a cycle has been built.

After a cycle was found, pheromones on the edges contained in the cycle are increased based on the length of the resulting cycle such that a smaller length results in a higher number of pheromones.

While optimizing, the pheromones on the edges steadily decay.



Ant Colony continuously improves the solution and most solutions are good and fast found.

Genetic Algorithm

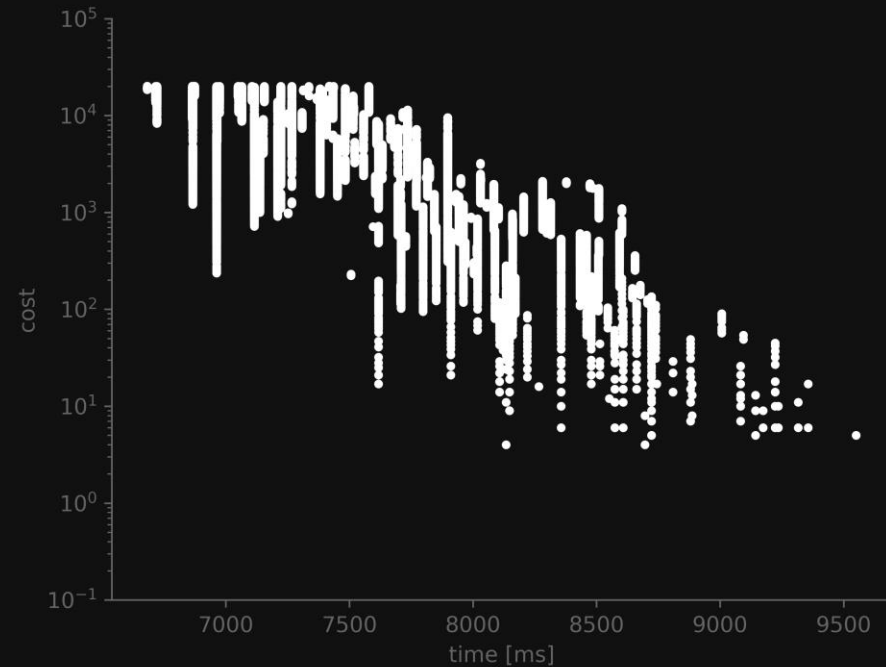
Every individual is a permutation of nodes and in each generation, we mutate and crossover the individuals.

Mutations are swaps of two nodes in the permutation sequence of a given individual.

Crossovers transfer path segments (permutation sequence parts) and the destroyed parts are fixed to recreate the cycle.

The best quarter of individuals is used for crossovers with the others.

The best individual is always kept as is.



GA (with crossover) is **unstable** - Depending on the initial Permutation, **loss** can **decrease fast** or **stay high** for a long amount of time. The **quality of solution can vary heavily**.

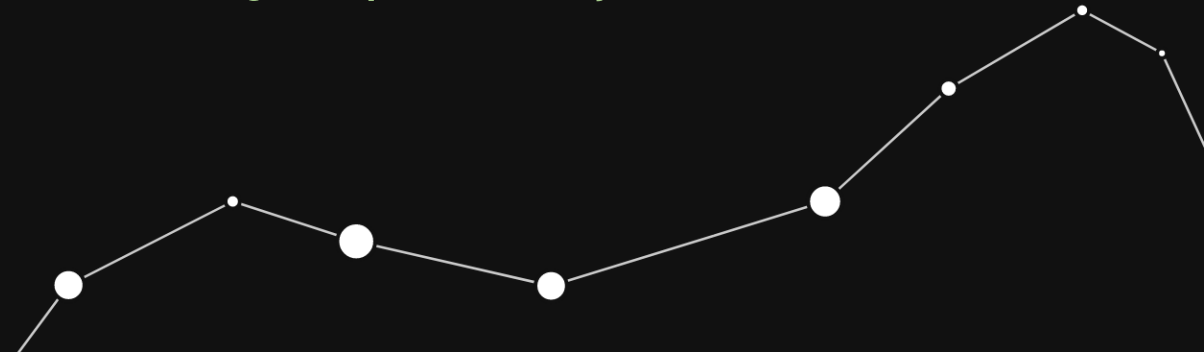
Conclusions

The **Bruteforce** approach is guaranteed to find the optimal solution but performs bad as the number of possibilities becomes too large even for a small number of nodes.

The **Genetic Algorithm** approach seems to struggle with crossovers, since the combination of two different cycles will lead to a destroyed cycle in most cases making it difficult to converge to a good solution.

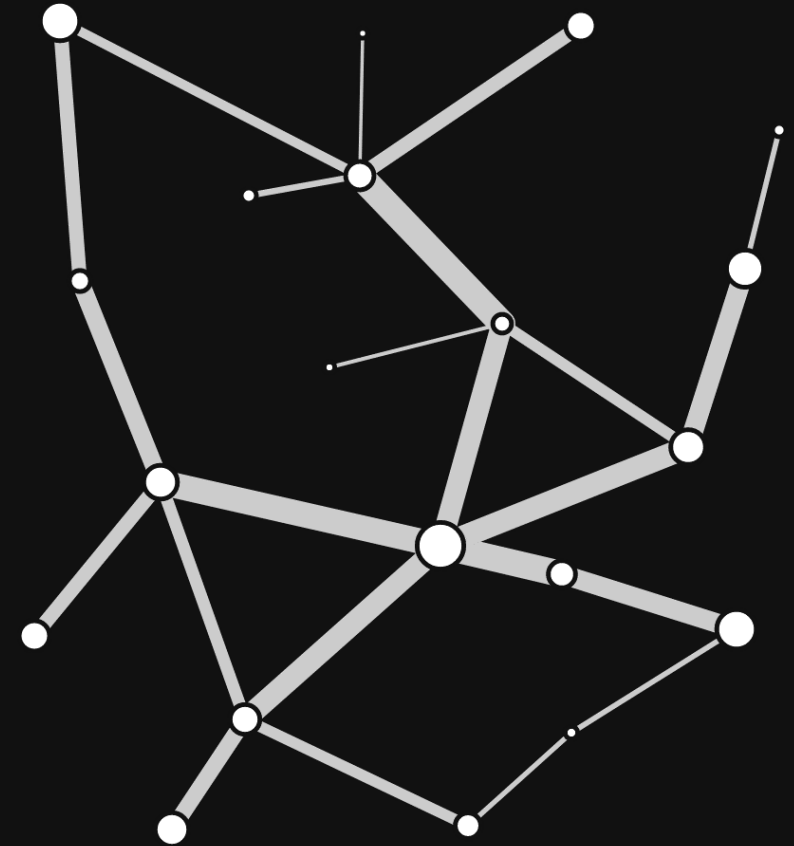
Simple swaps of nodes in the permutation, as in **Simulated Annealing** (steps) or **Genetic Algorithm** (mutation), seem not strong/effective enough to find good solutions.

Ant Colony performs the best due to the collaboration of the ants via pheromones, which enables selecting good edges with a higher probability.



Traffic Problem

We aim to find the set of streets (edges) connecting all cities (nodes), which **minimizes the overall cost for building** these streets.



Routing

Every city has a fixed population of people, which is distributed to all other cities using their population as a weight. Large cities will attract many people, while smaller ones will attract proportionally less.

The amount of people travelling from one city to another is then accumulated as traffic on the edges of the shortest path between these cities. This gives us the final amount of traffic on every edge, after doing this for all cities.



Cost

For every edge contained in the set of used edges there are two costs:

- The first one is the initial cost c_{init} for building that edge, which is simply a factor α .
- The second one is the traffic cost $c_{traffic}$, which increases with the length and the amount of traffic routed over that edge and is multiplied by a factor β .

For edges not contained in the set of used edges the cost is zero.

$$c(E_{used}) = \sum_{e \in E_{used}} c_{init}(e) + c_{traffic}(e)$$

$$c_{init}(e) = \alpha$$

$$c_{traffic}(e) = \beta \times length(e) \times traffic(e)$$



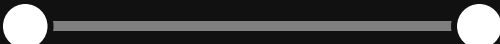
$$c(e) = 0 + 0$$



$$c(e) = 100 + 20$$



$$c(e) = 100 + 100$$



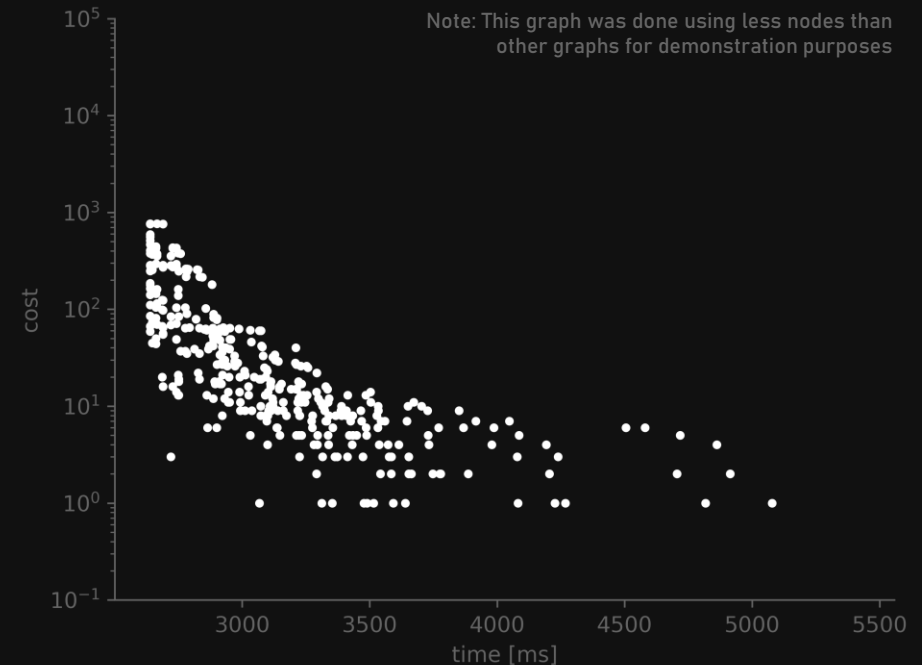
$$c(e) = 100 + 200$$

Bruteforce

Again, the Bruteforce approach tests all possible sets of edges and takes the best.

Assuming we have n nodes, there can be a total of $\frac{n(n-1)}{2}$ edges in the graph. So, if all those edges can be either in the set of used edges or not, we have a total of $2^{\frac{n(n-1)}{2}}$ possibilities.

With the resources needed for the traffic and cost calculation this **becomes computationally infeasible even for small n** (in our implementation for $n > 8$).



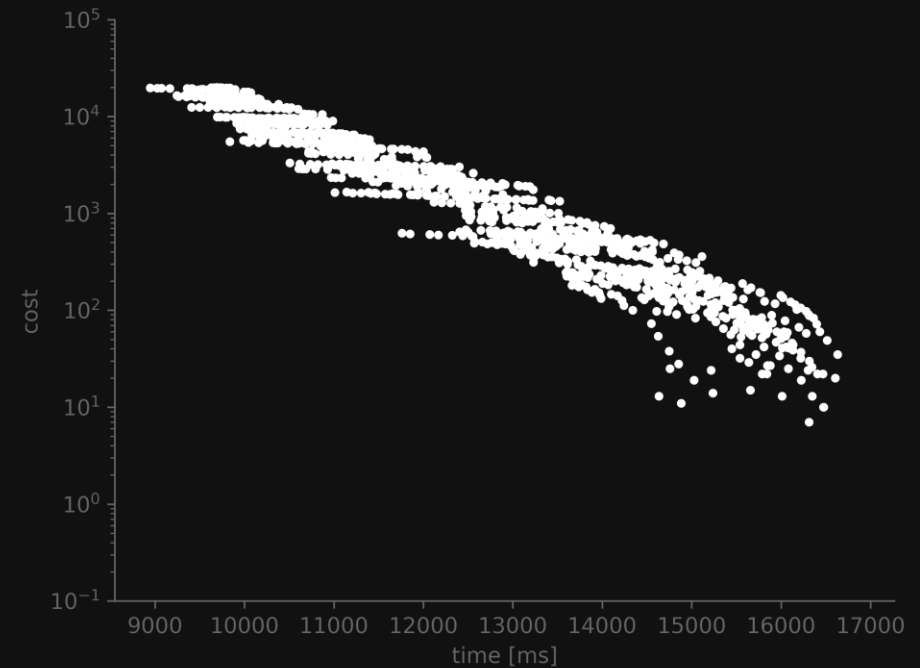
Bruteforce is **guaranteed to find the optimal solution** but has **no guarantees on the time required** to do so.

Simulated Annealing

The implementation is similar to the one from SA in TSP (and lecture).

However, this time the neighborhood is defined as the set of graphs using an edge more or less.

Therefore, steps are enabling or disabling edges, while keeping the graph connected.



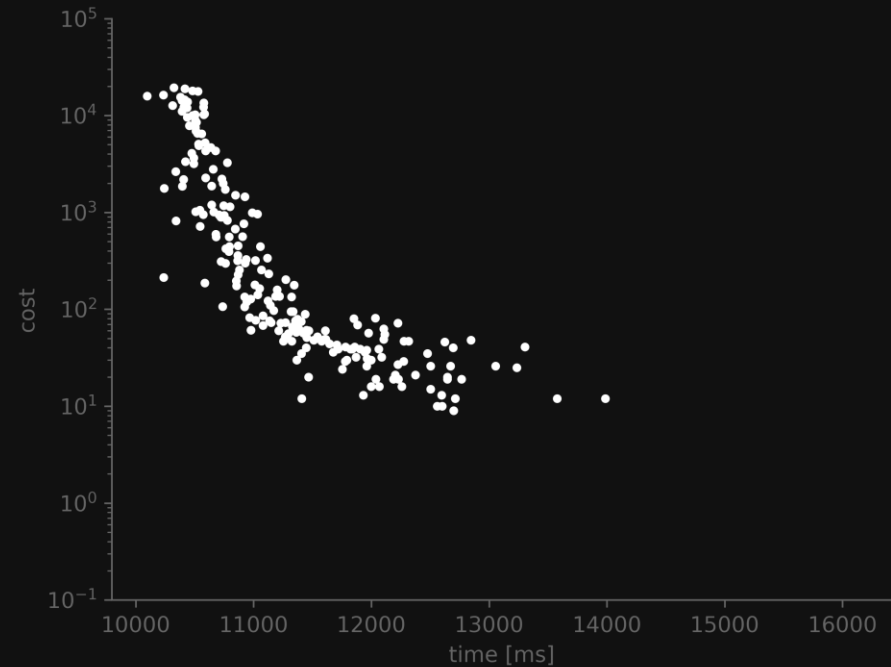
Simulated Annealing **steadily improves the solution** but is **slow and can get stuck rather easily**

Ant Colony

In every iteration ants will randomly select a set of edges (connecting the whole graph) depending on the number of pheromones on the individual edges.

After selecting a possible set, the ant will calculate the traffic and cost and increase the pheromones on the edges accordingly, such that a low cost means a high number of pheromones.

Over time the number of pheromones on the edges will steadily decay.



Ant Colony finds a "good" solution fast but then has trouble improving that solution further

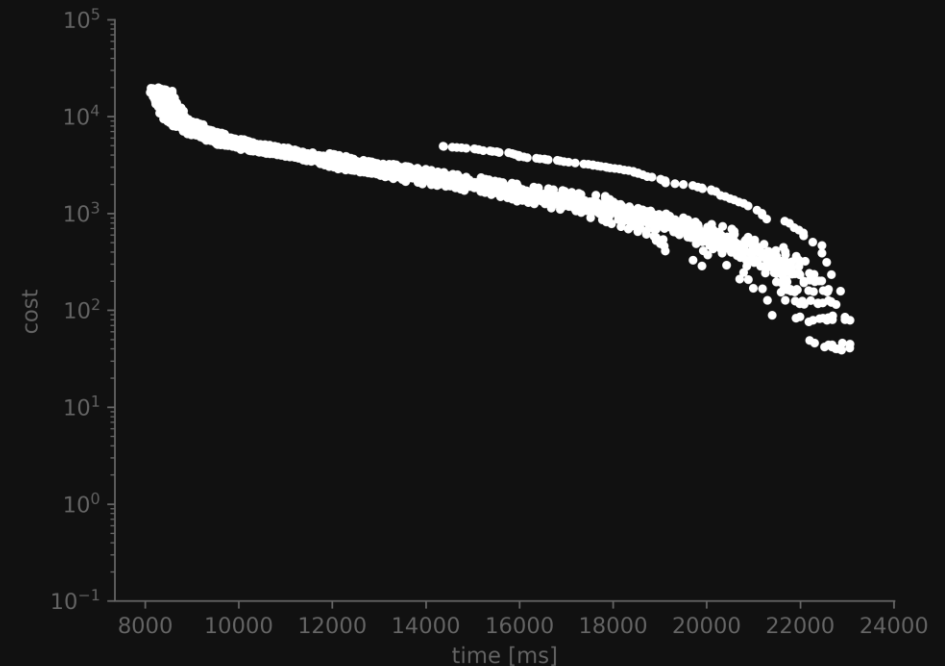
Genetic Algorithm

Every individual represents a set of used edges (streets) on the given graph.

Mutations either disable or enable an edge, without destroying connectivity of the graph.

Crossovers combine the sets of edges from two individuals by randomly taking parts from both sets.

Otherwise, the algorithm is similar to the TSP problem. In each generation we do mutation and crossover with the quarter fittest individuals.



GA improves very smoothly and even speeds up considerably since it starts from a complete graph and has a lot of computational effort in the beginning.

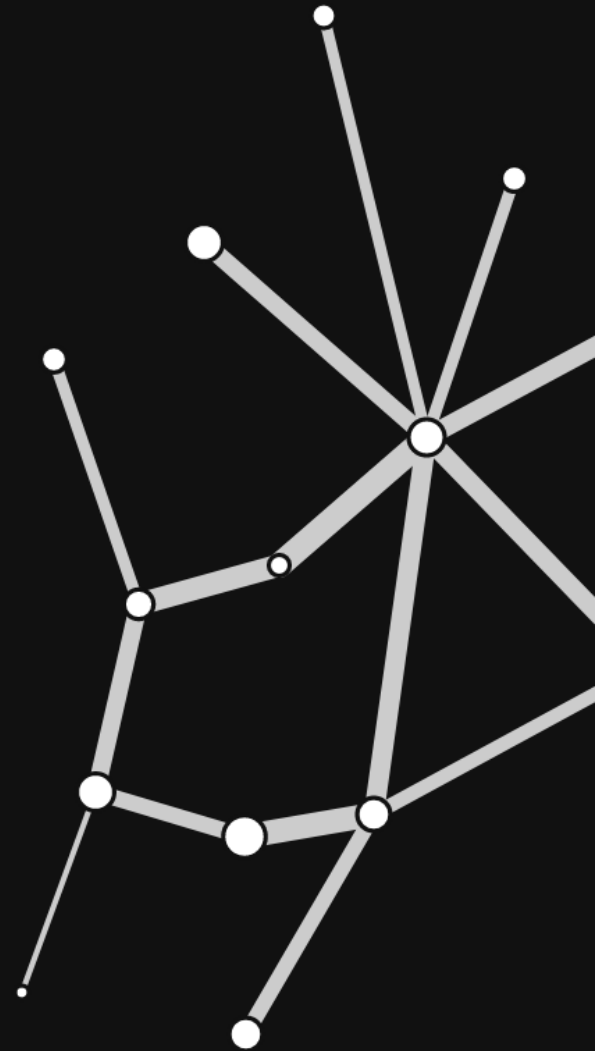
Conclusions

Like with the TSP, Bruteforce **becomes infeasible** even for a small number of nodes (even less than for the TSP).

Genetic Algorithm **works best for this problem**. Unlike in the TSP problem, the **crossovers seem to help GA** converging and don't work against it.

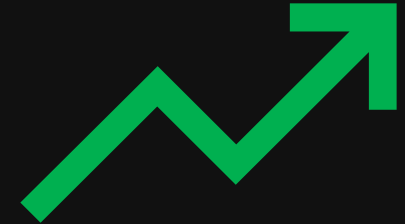
Due to the heavy computational load of the Traffic Problem, we have to be careful to **choose low enough population sizes** for **Genetic Algorithm**. While bigger populations would probably result in better convergence, they make the whole algorithm increasingly slow.

Increasing step sizes in SA and mutation probability in GA **can speed up convergence**. However, choosing too high values **can make the algorithms unstable**. Thus, finding perfect parameters is an art in itself and differs from graph to graph.



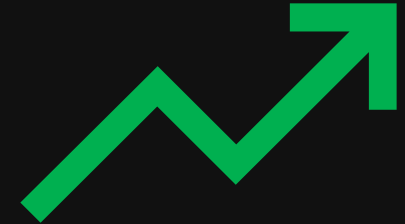
Possible Improvements

- Restrict the search space to edges between close cities, such that edges between far away cities are ignored, since they aren't usually part of the solution.
- Allow fixation of edges, such that one could optimize around an already existing networks of roads.
- Introduce barriers, such that roads need to be built around them or through them at a higher cost (e.g., tunnels or bridges).
- Make traffic routing more realistic:
 - Route people to closer cities with a higher probability, since people tend to commute to closer cities, rather than to far away ones.
 - Distinguish between city population and attractiveness, since a city may be small in population, but attract a lot of people.



Possible Improvements

- Don't start from a complete graph for **Genetic Algorithm** or random graph for **Ant Colony**. Maybe start from a random tree.
- Find a better way to deal with connectivity – Currently unconnected graphs get discarded after a connectivity check and generating new connected graphs is required, which can impact performance.
- Employ random restarts for **Simulated Annealing** to avoid bad starting points.

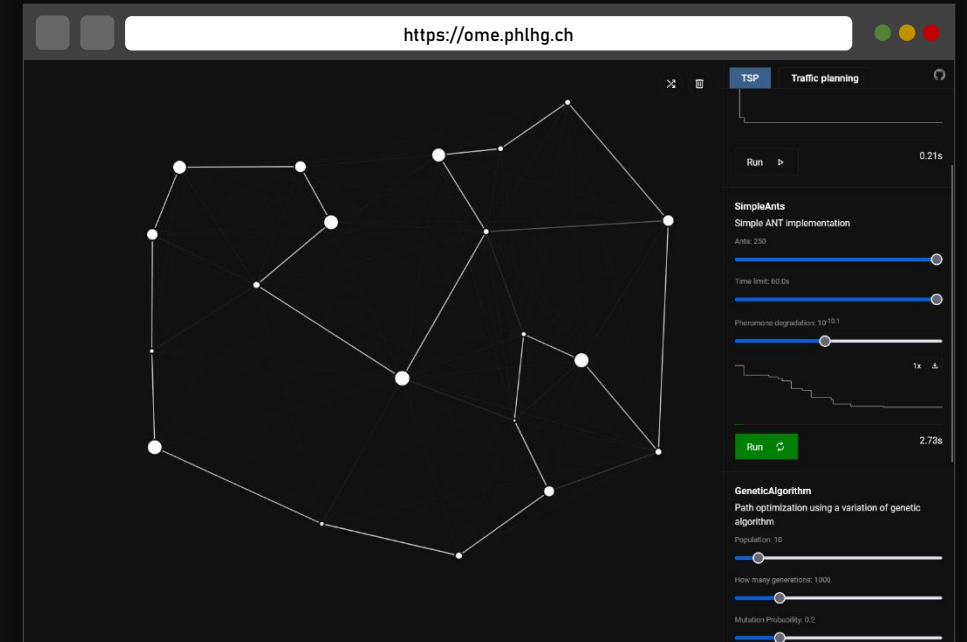


Website & Source Code

Implementation

Website with an interactive implementation:
<https://ome.phlhg.ch/>

Source Code of Web Application & Analysis:
<https://github.com/phlhg/traffic-sim>



Philippe Hugo: Web Application, Design, Problem, Bruteforce, Slides
Aaron Hodel: Ant Colony
Philipp Scherer: Genetic Algorithm, Simulated Annealing, Slides