MA4254

Computational Assignment

Alon Rafael Landmann

A0276620E

May 17, 2024

1 Facilities Allocation

1.1 Introduction and prior analysis

In the facility location problem, we are tasked with finding the optimal way to serve a number of customers via a given set of facilities. In our analysis of the problem, 25 customers and 15 facilities have been randomly distributed on the unit square $[0, 1] \times [0, 1]$.

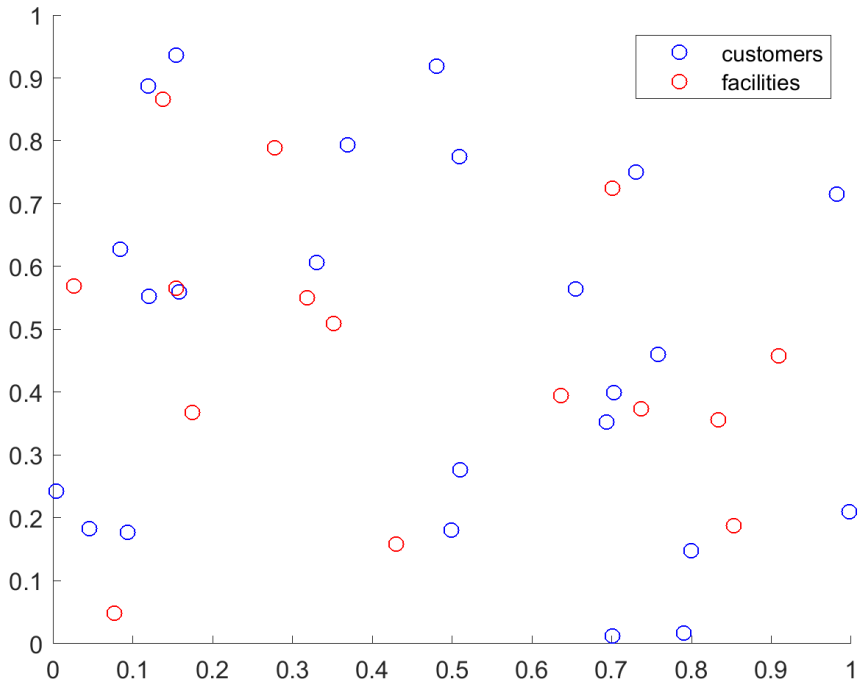

Figure 1: Map of facilities and customers.

Whenever a facility is tasked with serving at least one customer, it must be "switched on" at a cost of c_j, where $j = 1, \ldots, n = 15$. Additionally, serving customer i with facility j comes with a cost d_{ij} ($i = 1, \ldots, m = 25$). In our implementation, this cost will be exactly equal to the distance between the customer and the facility, and c_j will be 1 for all facilities.

The Facilities Location Problem seeks to minimize this cost while serving every customer. We can express this problem as a MILP. Let y_j be the binary variable which is 1 if facility j is switched on, and let x_{ij} model whether customer i is served by facility j.

$$\begin{aligned} \min & \sum_{j=1}^{n} c_j y_j + \sum_{j=1}^{n} \sum_{i=1}^{m} d_{ij} x_{ij} \\ \text{s.t.} & \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\ & x_{ij} \leq y_j \quad \forall i, j \\ & 0 \leq x_{ij} \leq 1, y_j \in \{0, 1\} \end{aligned}$$

(c) This is the FLP. We can form its linear relaxation (FLP-LR) by dropping the integer constraints.

$$\min \sum_{j=1}^{n} c_j y_j + \sum_{j=1}^{n} \sum_{i=1}^{m} d_{ij} x_{ij}$$
$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i$$
$$x_{ij} \leq y_j \quad \forall i, j$$
$$0 \leq x_{ij}, y_j \leq 1$$

The following formulation is an equivalent formulation to FLP, known as the aggregate facility location problem (AFL).

$$\min \sum_{j=1}^{n} c_j y_j + \sum_{j=1}^{n} \sum_{i=1}^{m} d_{ij} x_{ij}$$
$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i$$
$$\sum_{i=1}^{m} x_{ij} \leq m y_j \quad \forall j$$
$$0 \leq x_{ij} \leq 1, y_j \in \{0, 1\}$$

(d) We can also give its linear relaxation, which we will denote by (AFL-LR).

$$\min \sum_{j=1}^{n} c_j y_j + \sum_{j=1}^{n} \sum_{i=1}^{m} d_{ij} x_{ij}$$
$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i$$
$$\sum_{i=1}^{m} x_{ij} \leq m y_j \quad \forall j$$
$$0 \leq x_{ij}, y_j \leq 1$$

(a) Note that while the original FLP formulation has $m \times n$ forcing inequalities (beyond the variable bounds), the AFL formulation only has n inequalities, one for each facility.

(b) We can show that the FLP and the AFL formulations are equivalent.

Proof 1 *To show that the two formulations are equivalent, it suffices to show that the feasible regions are the same. This is because the two formulations have the same objective function.*

$\Longrightarrow$: *Let (x, y) be a feasible solution to the (FLP) formulation. We need to show that the following new constraint is satisfied.*

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \quad \forall j$$

Fix a facility j. If all x_{ij} for this j are equal to zero, then the left hand side of the inequality is zero, and hence, no matter what y_j is, the inequality holds. If for some i x_{ij} is not equal to zero, then by the constraint

$$x_{ij} \leq y_j \quad \forall i, j$$

y_j is forced to be equal to 1, and hence the right hand side is equal to m. The left hand side is clearly bounded by m as there are m terms - all of them bounded above by 1

$\Longleftarrow$: *Let (x, y) be a feasible solution to the (AFL) formulation. We need to show that the following new constraint is satisfied.*

$$x_{ij} \leq y_j \quad \forall i, j$$

Fix and i and a j. If x_{ij} is equal to zero, then the inequality holds as $y_j \in \{0,1\}$ is given. If x_{ij} is not equal to zero, then in the constraint below, we know that the left hand side for this j has to be larger than zero

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \quad \forall j$$

Hence, y_j cannot be zero, hence it must be 1. Therefore, the constraint $x_{ij} \leq y_j$ is also satisfied.

(e) Before we run our implementation, we can speculate on the optimal values of these 4 formulations. Since the feasible regions of the linear relaxations are supersets of their respective MILPs, we can conclude that the optimal values of the relaxations must be equal to or less than (i.e. better) than the original formulations. Furthermore, since the original FLP and AFL formulations are equivalent, we know that their optimal values must also be the same. However, it is not possible or trivial to draw prior conclusions about the relationship between the two linear relaxations, or whether or not the relaxations are going to be tight or not. The proof that the two original formulations were equivalent was relying on the fact that the facility variables were binary variables. We do not have this fact for the relaxations, and so we cannot draw similar conclusions.

Before moving on to some implementation details, we give the formulation of the capacitated FLP problem (C-FLP). Below, r_j represents the maximum number of customers facility j can serve. In our implementation, this will be 2 for all facilities.

$$\begin{aligned}
&\min \sum_{j=1}^{n} c_j y_j + \sum_{j=1}^{n}\sum_{i=1}^{m} d_{ij} x_{ij} \\
&\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\
&\sum_{i=1}^{m} x_{ij} \leq r_j \quad \forall j \\
&x_{ij} \leq y_j \quad \forall i, j \\
&0 \leq x_{ij} \leq 1, y_j \in \{0,1\}
\end{aligned}$$

The linear relaxtion of this formulation (C-FLP-LR) is given, too.

$$\begin{aligned}
&\min \sum_{j=1}^{n} c_j y_j + \sum_{j=1}^{n}\sum_{i=1}^{m} d_{ij} x_{ij} \\
&\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\
&\sum_{i=1}^{m} x_{ij} \leq r_j \quad \forall j \\
&x_{ij} \leq y_j \quad \forall i, j \\
&0 \leq x_{ij}, y_j \leq 1
\end{aligned}$$

1.2 Implementation

The MILPs introduced in the first subsection are solved in MATLAB using the "intlinprog" function provided by the global optimization toolbox. The function takes objective function coefficients, two linear systems - one for inequalities, and one for equalities - each represented by a coefficient matrix A and a right-hand-side vector b, as well as lower, and upper bounds for the variables, and a vector denoting which variables are to be constrained to $\mathbb{Z}$. Default settings for integer tolerance were used.

In generating the appropriate matrices and vectors, the arrangement of the decision variables had to be determined and kept consistent. The first n locations were allocated to the y_j variables, and the remaining $m \times n$ slots were alloced to the x_{ij} variables. The order observed thereby was to first take all the x variables belonging to facility $j = 1$ cycling through all the customer indices, and then to move on to $j = 2$, and so on.

Then, a loop was created. In each iteration, a new map was generated randomly, and then all 6 formulations were run, and the results stored.

1.3 Results

The following plot displays the results of the analysis. Each problem formulation is represented by a different marker. The MLP formulations are marked by a circle, and the linear relaxations are marked by a cross. The colors distinguish the type of formulation.

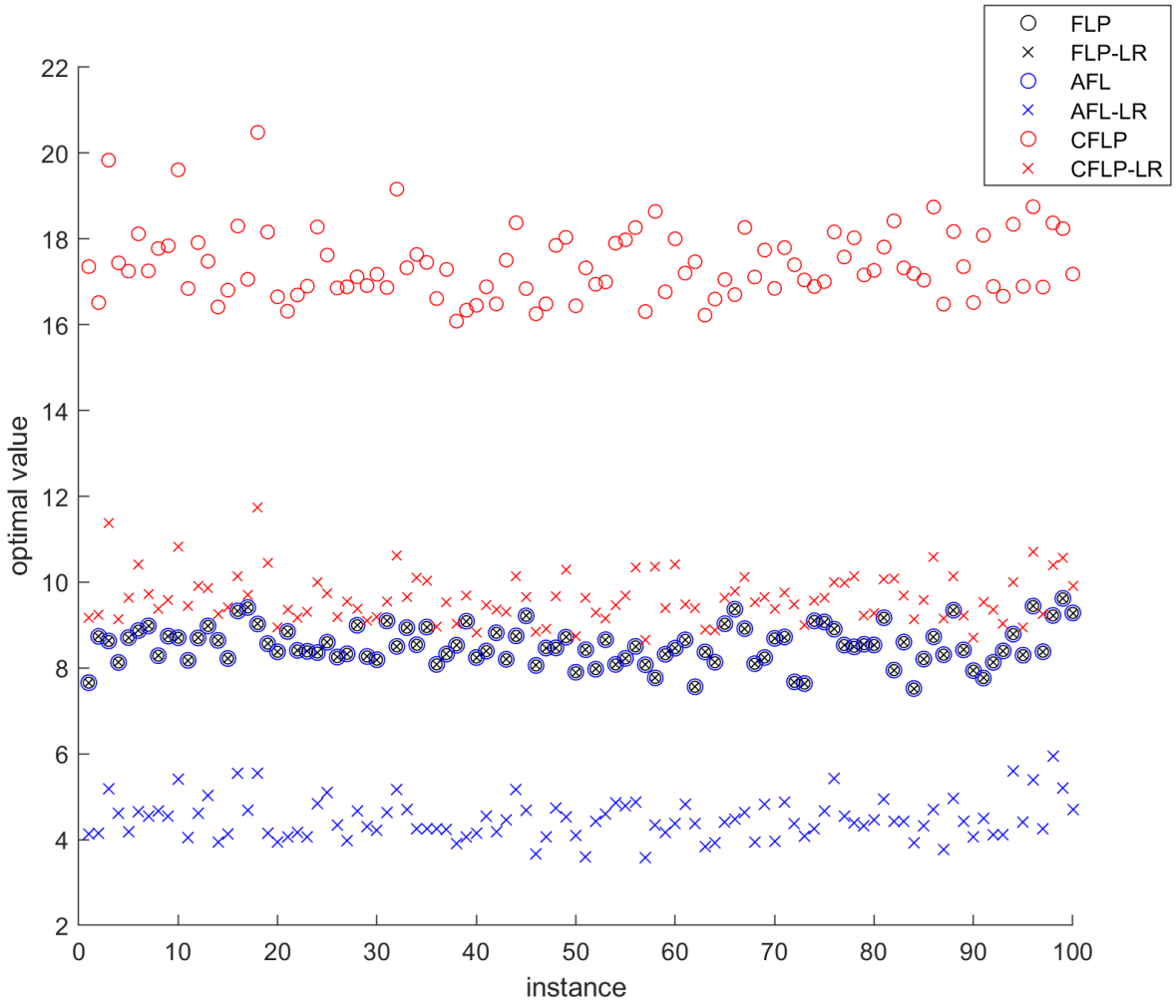

Figure 2: Map of facilities and customers.

We can see that the (FLP) formulation is almost always tight, that is, the optimal value found for the original formulation is equal to the value found for the linear relaxation. In fact, only three instances showed a difference above the tolerance of 10^{-10}.

We can also see that, as predicted, the optimal values for the FLP and the AFL formulations agree precisely. The proof for this result has been given.

However, we can note that for the (AFL) formulation, the linear relaxation is not tight anymore. This formulation has fewer inequalities, and the only reason it properly modelled the facility forcing constraint, was that the facility variables were binary variables. In this relaxation, facilites can be "partially switched on", and a new solution with lower cost can be found in which customers are served by multiple locations "in parts".

Finally, we can see that the capacitated problem clearly has a higher total cost than the uncapactitated problem. This is to be expected, as of course, one facility servicing multiple customers that were close to it, was a highly efficient allocation in the original problem, which was not allowed in the C-FLP problem.

The linear relaxation of this formulation is also never tight.

2 Travelling Salesman

2.1 Introduction and prior analysis

The travelling salesman problem features a distribution of cities, and a salesman who is to go on a closed tour visiting every city exactly once and returning to the city he started at. The objective is to find the shortest possible tour.

The associated map in our implementation is again the unit square $[0, 1] \times [0, 1]$, and we will have 50 cities distributed at random, similar to how customers or facilities have been distributed in Figure 1.

The travelling salesman problem is notoriously difficult to solve. There search space is vast as there are $(n - 1)!$ different possible tours (if we do not distinguish tours by which city is the starting city). A potential plethora of local optima exist which may not be global optima.

Here, we are going to use a simulated annealing algorithm to approach the problem.

2.2 Implementation

The simulated annealing algorithm is inspired by the annealing process in metallurgy. When metals are worked in the liquid phase and then cooled, the final properties of the material are highly dependent on what temperature was reached and, critically, how quickly the metal was cooled. The more time it takes for the metal to cool, the more time there is for individual crystals to form, and thus, the metal will have an average crystal size that is larger than if it were cooled more rapidly. Its mechanical properties will obviously depend on this factor.

In the simulated annealing algorithm, a global temperature T is given, which is exponentially decreased at every iteration according to a predetermined cooling rate η.

The algorithm starts with a random solution vector, a tour in our case. During each iteration, the currently preferred tour is matched against a slightly perturbed version of the same tour. If the new tour has a shorter length, it replaces the old one right away. However, even if it is a worse solution with a longer length, in order to explore the search space for more global optima, and to tunnel out of and away from potentially bad local optima, the algorithm still gives a chance to replace the currently preferred tour by the new tour. This choice is done according to the following formula.

$$\exp(-\frac{f_{cand} - f_{curr}}{T}) \geq u,$$

where f_{cand} and f_{curr} are the lengths of the new candidate tour and the current tour respectively, and u is a variable drawn uniformly from $[0, 1]$.

From the above formula, we see that the bigger the difference between the worse suggestion and the current tour, the smaller the chance that it will get swapped in. However, a high temperature T will much improve this chance, and allow for more exploration of the search space.

A lower global temperature will, on the other hand, discourage curiosity, and let the algorithm sink into a local minimum.

Figure 3 clearly shows these two different phases of the algorithm. The red line marks the global temperature on a logarithmic scale. As long as the temperature is roughly above $10^0 = 1$, the algorithm is exploring the search space quite freely. It then, rather quickly, sinks into a realm where it doesn't have enough energy anymore to explore anything that is much worse than its current choice, and begins the new phase, in which it simply descends towards the local minimum.

Do note that there is nothing special about the breaking point being reached around 1 in this process. Where this breaking point lies depends on the general magnitude of the objective function values.

In our implementation, we have tested different starting temperatures, ranging all the way from 10^{-50} up to 10^{50}, and 10'000 iterations were performed with a cooling rate of $\eta = 0.99$.

In addition, two different mechanisms for generating perturbed tours were used. Let a tour be given by a permutation m of the integers $1, \ldots, 50$. The first mechanism chose two indices $i < j$ at random and flipped the segment $m_i, \cdots, m_j$ front to back. The second mechanism simply swapped the elements m_i and m_j of the permutation, where the indices were again chosen randomly.

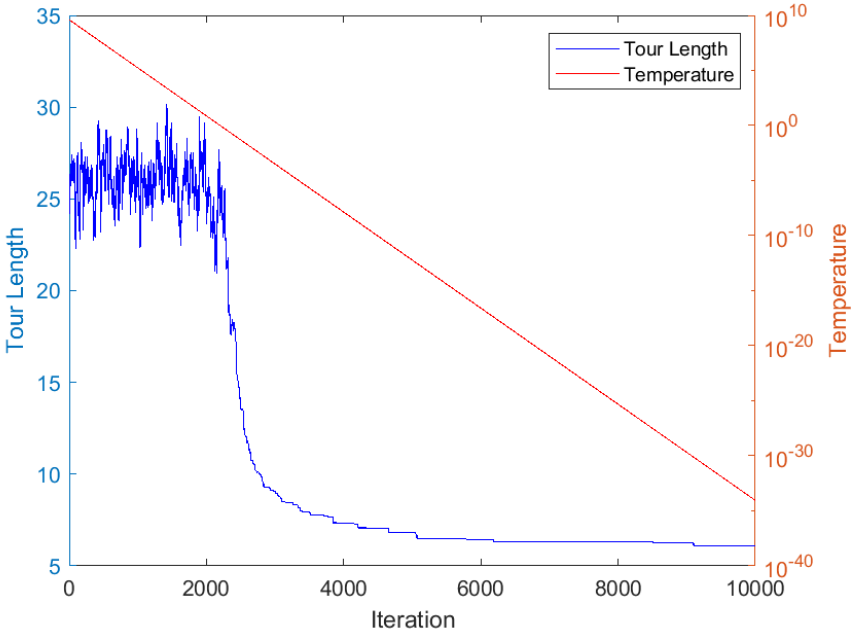

Figure 3: Simulated annealing algorithm with $T_0 = 3.9442 \cdot 10^9$.

2.3 Results

Figure 4 displays the overall results of the algorithm. The horizontal axis marks the initial temperature used, and the vertical axis shows the length of the tour preferred at termination of the algorithm. We can see two sets of data, one for each perturbation method.

Do note that the same map of cities has been used accross all of these simulations.

The first thing to notice, is that, overall, the length of the best path found is quite stable across a wide range of initial temperatures. We will discuss this more when we look at some individual examples.

For now, we do have to mention, that, although the value remains quite stable, it is not exactly equal. This implies that many different "good but not perfect" tours exist and have been found. We will also see these later.

It is apparent that for this problem, the flipping perturbation method has been shown to work more efficiently than the swap perturbation. We can speculate that is is because a flip is all in all a stronger perturbation than a swap, as more indices get changed. This should allow for a more comprehensive and rapid exploration of the search space.

Finally, we should address the spike that can be seen for the higher initial temperatures. This spike means that the algorithm terminated with a bad tour. As we will see in the examples that follow, this is because the algorithm did not reach the second phase, the cooler one, in which it descends into the local minimum.

Let us now have a look at some individual instances with different starting temperatures.

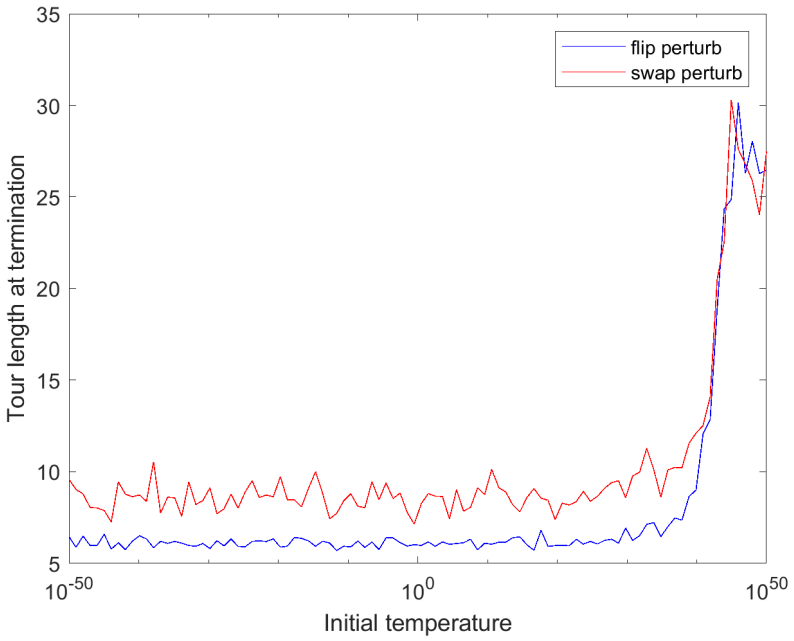

Figure 4: Simulated annealing of TS Problem results. $k_{max} = 10000, \eta = 0.99$.

We can see that these examples (Figures 5 - 11) perfectly illustrate the patterns discussed in the overall analysis. If we look at figure 7, we again clearly see the two phases working well in tandem with each other. First, the algorithm explores the whole search space, and then has a lot of time to settle into a local minimum. The associated solution shows no intersections, which is a good sign.

If we increase the initial temperature, we can see that this elongates the exploratory phase, as the critical temperature where the shift happens is delayed. This has the effect that the second phase is shortened, and this results in the local minimum being less explored, and thus less optimized. At an initial temperature of approximately 10^{30}, in Figure 9, we can see that the second phase is cut a bit too short and the tour at termination has some crossovers. Other inefficiencies such as zig-zag maneuvers have also been detected for these initial temperatures during other trial runs.

In Figure 10, at an initial temperature of roughly 10^{40}, we reach a point, where we have quite clearly not had enough time to optimize locally, and the resulting tour is clearly sub-optimal - we are beginning to climb the spike of Figure 4.

Finally, Figure 11 illustrates what happens when the global temperature remains too high during the entire run-time and the algorithm terminates during the exploratory phase. The result is a completely chaotic and

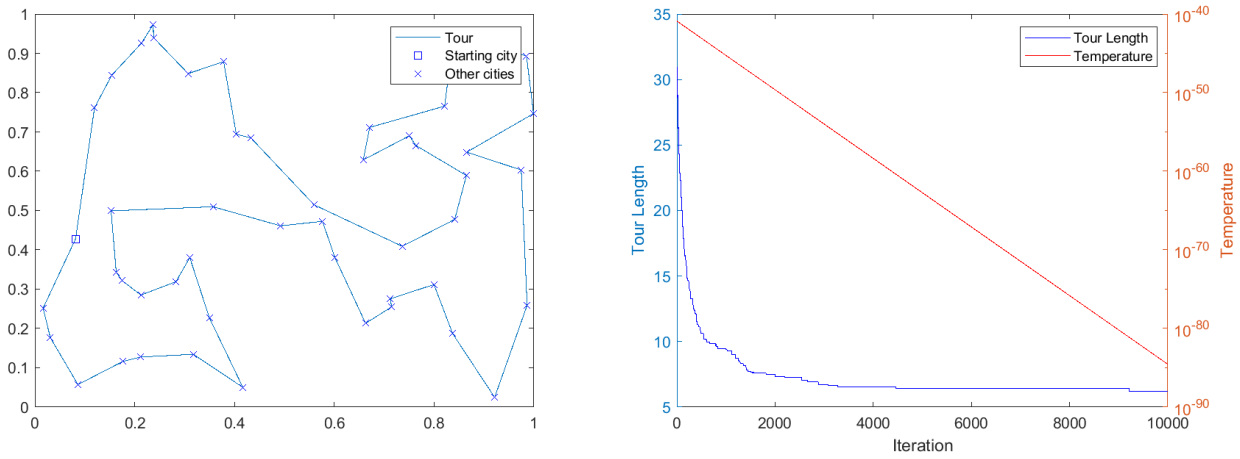

Figure 5: Flipping perturbation, $T_0 \approx 10^{-40}$

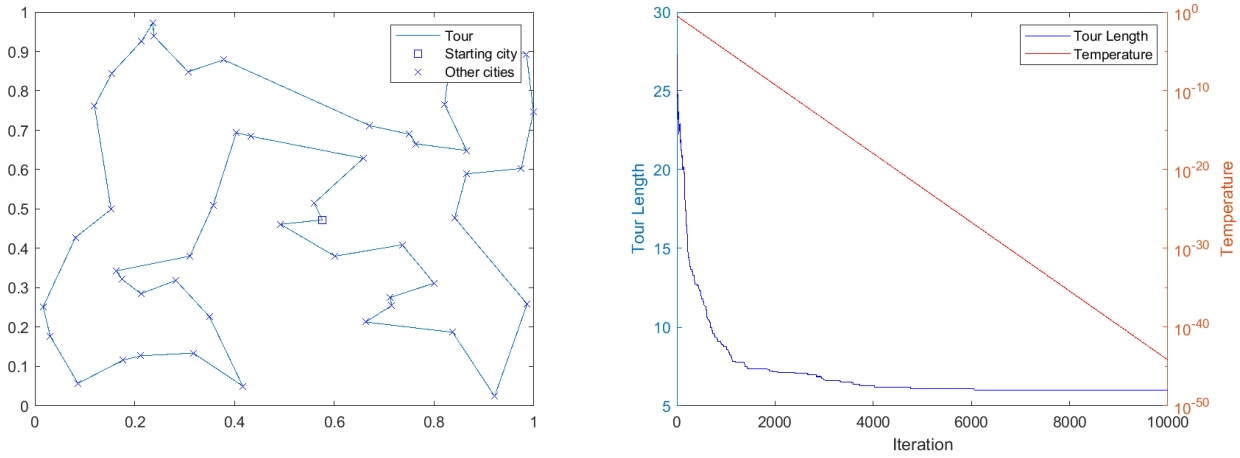

Figure 6: Flipping perturbation, $T_0 \approx 10^0$

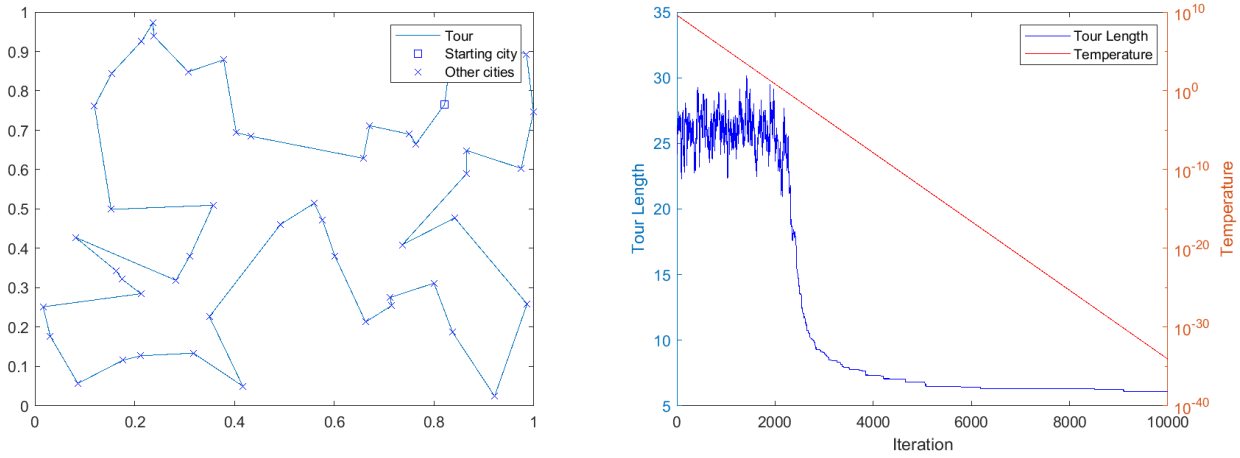

Figure 7: Flipping perturbation, $T_0 \approx 10^{10}$

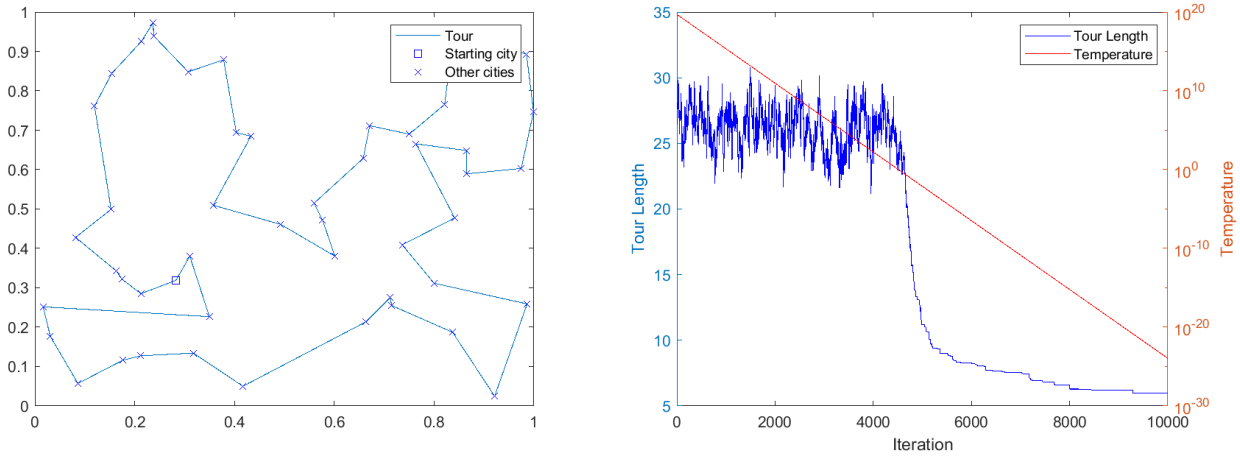

Figure 8: Flipping perturbation, $T_0 \approx 10^{20}$

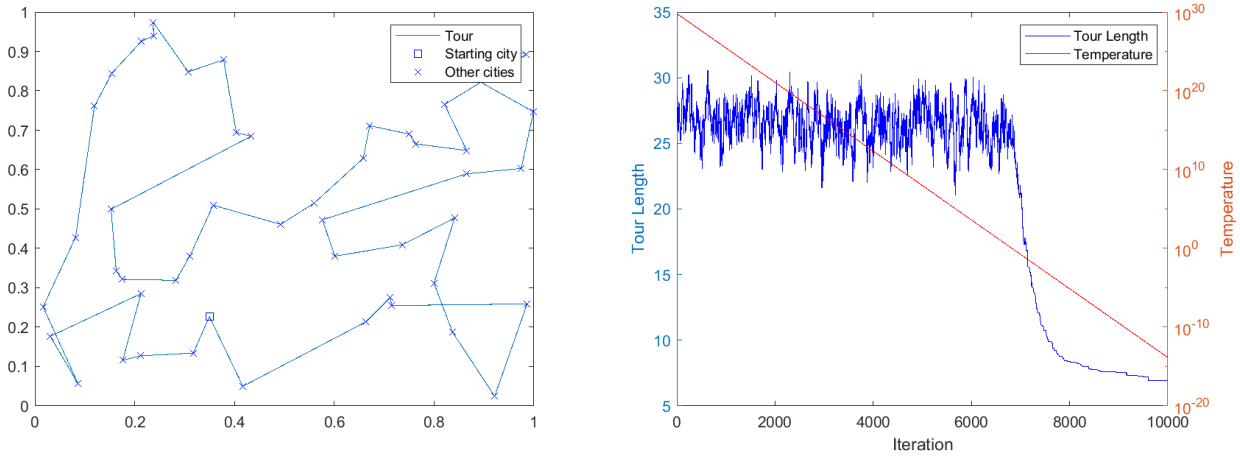

Figure 9: Flipping perturbation, $T_0 \approx 10^{30}$

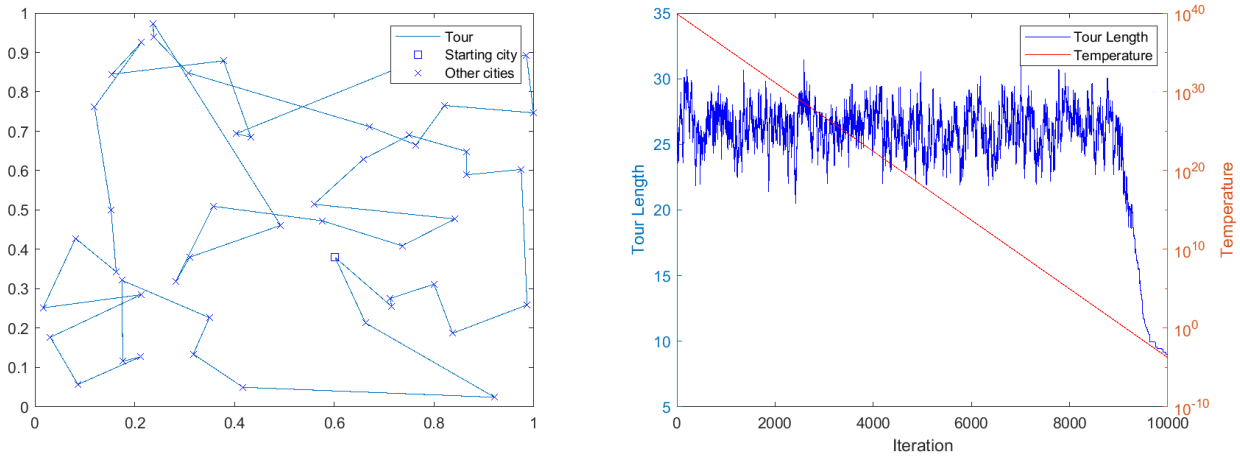

Figure 10: Flipping perturbation, $T_0 \approx 10^{40}$

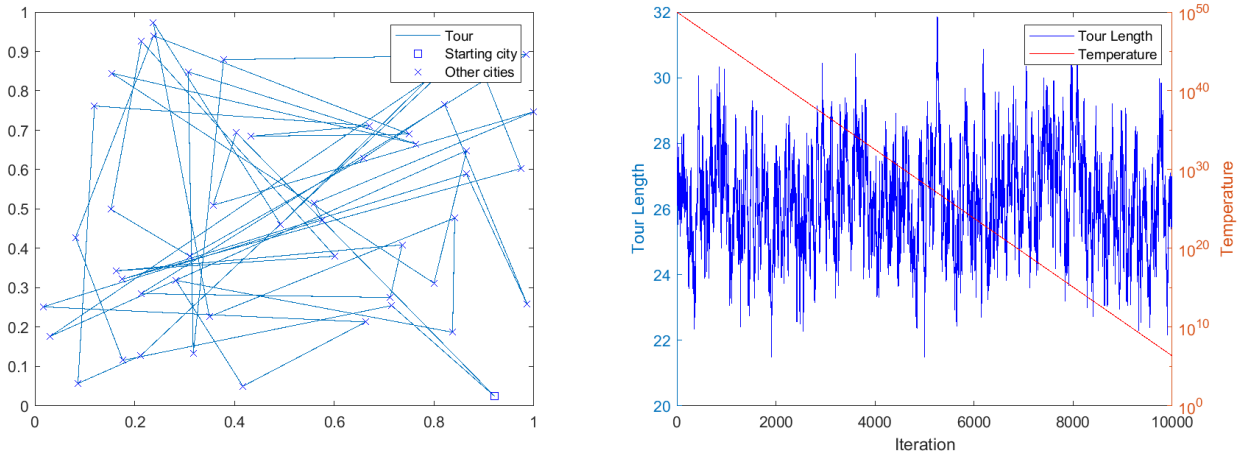

Figure 11: Flipping perturbation, $T_0 \approx 10^{50}$

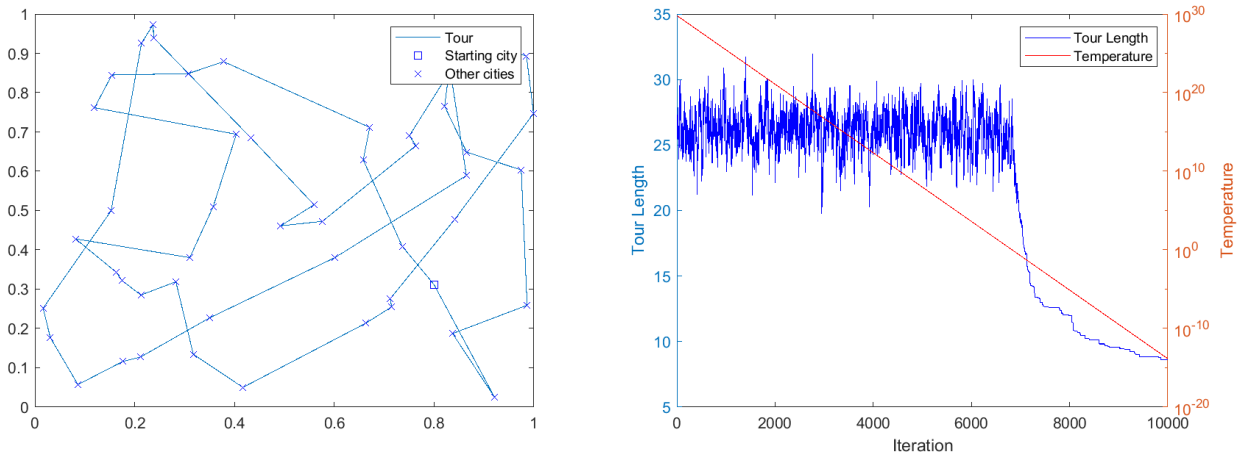

Figure 12: Swapping perturbation, $T_0 \approx 10^{30}$

inefficient tour.

Interestingly enough, if we look at Figures 5 and 6, we see that for very low initial temperatures, a good solution without cross-overs is also found. The results look similar for initial temperatures between those two. For these low temperatures, we can see that there basically is no exploration of the global space happening. Phase 1 of the algorithm is non-existent. The cool initial temperature forces descent into a local minimum very rapidly. However, this does not seem to be a huge problem for this particular problem. One possible explanation for this has been hinted at when we discussed that there seemingly are many different "good" tours with roughly equal total length. Having analyzed many of those good solutions without cross-overs, there has been quite a lot of variety among them, for the same set of cities. The overall shape of the path does not seem to be as crucial as solving for local efficiencies. This is certainly an interesting conclusion. There is likely only one unique optimal tour, and it must lie in one of these many local minima. But from our analysis, we have no hint as to which of them it might belong. This yet another sign that the TSP problem is very difficult.

Finally, we quickly have a look at one more set of graphs to show the difference the perturbation method can make.

Figure 12 demonstrates a pattern common for the swapping perturbation. If we compare it to Figure 9, we see that the swapping method simply doesn't allow the algorithm to descend as deeply into the local minimum as the flipping method. As eluded to before, this might be due to the fact that it explores the search space much more narrowly and slowly, than the flipping method.