1 Heuristics for the Traveling Salesman Problem

We consider the following problem. We want to visit all the nodes of a graph as fast as possible, visiting each node exactly once and finally returning to the starting node. One practical example in the field of computer science is a computer network with one background process that is supposed to cyclically visiting all computers in order to distribute information or to synchronize them.
The classical motivating example is the problem of a traveling sales person who wants to organize the visits of his customers as efficiently as possible. To model this as a graph problem, one can use the cities of the customers as nodes, and the travel distances between each pair of cities as weighted edges between them. Then there exists (at least) one round trip through all nodes, visiting a node exactly once, minimizing the total travel distance of the trip.
Formally we define:

Definition 1 (Hamiltonian cycle) *A Hamiltonian cycle in a graph $G = (V, E)$ is a cycle visiting each node of V exactly once.*

The Traveling Salesman Problem can then be defined as follows:

Definition 2 (Traveling Salesman Problem (TSP)) *Given a complete graph $G = (V, E)$ with n nodes, and a function $c : E \to \mathbb{R}^+$ assigning positive costs to each edge. The problem is to find a Hamiltonian cycle in G with minimal edge costs.*

Because the graph is complete (there exists an edge between each two nodes), a Hamiltonian cycle exists in the graph and we only have to concentrate on finding such a cycle with minimal edge weight. In general graphs it is already hard to decide whether a Hamiltonian cycle exists.
Richard Karp proved in 1972 that the problem *"Given a graph $G = (V, E)$, does G contain a Hamiltonian cycle?"* is *NP*-complete (The definition and meaning of the term *NP*-completeness are discussed in the introductory lectures for theoretical computer science).
Without proof we state that also the Traveling Salesman Problem is *NP*-complete [1].
The consequence is: We cannot hope to devise an algorithm that finds the optimal solution in realistic computing time. We can only hope to devise algorithms giving good solutions.

[1]If you have previous knowledge in complexity theory: The proof works by reducing the decision problem *Hamiltonian cycle* to the decision problem *TSP*.

2 Nearest Neighbor Heuristic (NN) and 2-Opt

A simple method to find a round trip works as follows:

```
1. Start with an arbitrary city.
2. while (there are not visited cities left)
   Choose among the not visited cities the city $v$ that has the smallest distance
   to the current city. Add it to the tour and make it the current city.
3. Insert the starting city at the end of the tour to make it a round trip.
```

Figure 1: Nearest Neighbor Heuristic

It is easy to see that this approach yields some round trip, but not necessarily a good round trip in general. Often it is possible to improve the solution by simply exchanging two cities. Assume therefore that the cities $1, ..., n$ are visited in the following order:

$$\pi = \pi_1, ..., \pi_{j-1}, \pi_j, \pi_{j+1}, ..., \pi_{k-1}, \pi_k, \pi_{k+1}, ...\pi_n, \pi_1$$

In a so-called 2-Opt step, we choose two cities π_j and π_k and swap them in the sequence. The resulting round trip is then:

$$\pi' = \pi_1, ..., \pi_{j-1}, \pi_k, \pi_{j+1}, ..., \pi_{k-1}, \pi_j, \pi_{k+1}, ...\pi_n, \pi_1$$

By iteratively applying 2-Opt steps it is possible to generate new solutions and after applying “many” such steps we might have a substantially better solution than the one obtained by the Nearest Neighbor Heuristic.
The set of all solutions that can be obtained by applying a 2-Opt step to a given fixed solution π, is called the *2-Opt neighborhood* of π. (It is possible to extend this definition to the *k-Opt-neighborhood* of π, containing the round trips, that can be obtained from π by simultaneously exchanging k cities.)
Now we again consider only one 2-Opt step. Exchanging two cities of a round trip can make the resulting tour shorter, but it can also result in a longer tour. Is it better to only consider those 2-Opt steps that result in a shorter tour? Or would it be better to also sometimes consider steps that result in a longer tour? To answer this we take a look at the analogue of a hiking tour.
We are standing at the mountain top and want to descend to the valley (see Fig. 2). If we would only consider downward routes, we would possibly end our tour in a high mountain valley without getting to the actual valley. To be able to really go down, we sometimes have to go upwards, even though this leads a bit away from our goal.
The situation is very similar with several optimization methods. If we only consider improving steps, we will eventually end up with a solution that cannot be improved any more by only applying improving 2-Opt steps (local minimum). But this does not mean that we have reached the overall best solution (global minimum). Therefore, sometimes, it can make sense to apply a 2-Opt step even if the solution gets worse, because the

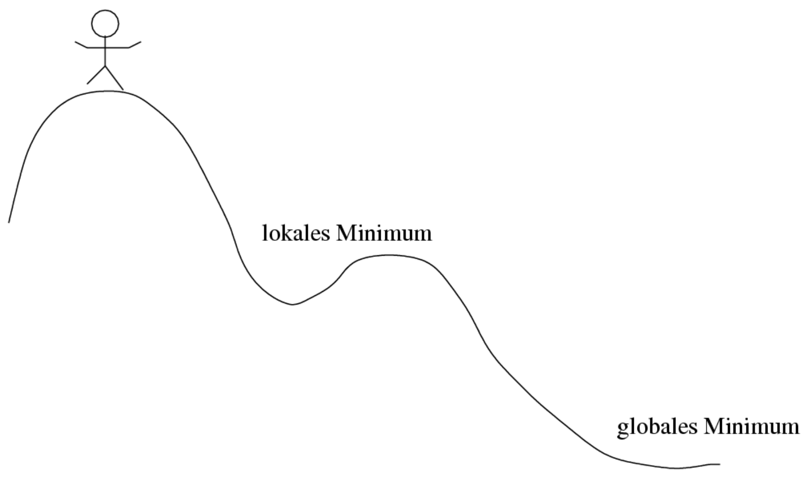

Figure 2: Descent to the valley

following steps may lead to a solution that is better in total. However, the improving steps should certainly be carried out more often, than the steps that worsen the solution. This idea is used in the scheme of *Simulated Annealing.*

3 Simulated Annealing

The term *Simulated Annealing* describes a class of algorithms for general optimization problems, including TSP.
Simulated Annealing tries to find the optimal solution in the set of all possible solutions and intents to reduce the probability to get stuck in a local minimum. This is achieved by also applying steps to worsen solutions, controlled by a randomized scheme. More precisely, a step from one solution x to another solution x' which is worse by Δ is accepted, if and only if

$$e^{(-\Delta/T)} > R,$$

where T is a control parameter and $R \in [0, 1]$ a random number.
The parameter T initially has a high value, so that many worsening steps are accepted. It is then slowly reduced to a low value, so that such steps are nearly always rejected.
The name of the method originates from an analogy to a process in thermo dynamics. If a material is heated above the melting point and then cooled down again, the resulting structure depends on the cooling rate. Bigger crystals for example are generated by cooling down slowly, whereas a fast cooling might lead to a high number of material defects. The simulation describes the energy change during the cool down phase until the system converges to a stable state.
Here the parameter T takes the role of the temperature, and it is important to develop a cooling scheme to decrease T. Additionally we have to define a start and end values for the temperature.

Altogether this results in the following algorithm:

```
1. Generate a starting round trip $x$.
2. Set $T$ to the initial temperature and set parameters $\alpha, k, l$.
3. while a break condition is not satisfied,
    (a) Apply a 2-Opt step and get solution $x'$.
    (b) Let $\Delta$ be the difference of the lengths of $x'$ and $x$.
        Case 1: $\Delta \leq 0$. Accept the new solution (set $x = x'$ ).
        Case 2: $\Delta > 0$.
            i. Generate a random number $R \in [0, 1]$.
            ii. Accept the new solution iff $e^{(-\Delta/T)} > R$.
    (c) After applying $k$ 2-Opt steps or $l$ improving steps set $T := \alpha T$ .
4. Output the current solution $x$ as the result.
```

Figure 3: Algorithm Simulated Annealing for TSP

4 Choice of parameters

When choosing the cooling scheme we have a lot of degrees of freedom and it is an art to tune the parameters to get a very good solution. Now we take a look at each parameter in more detail:

Initial temperature:

The process has to start such that most steps are being accepted, this means that the initial temperature has to be high. A lot of experience is necessary to quantify "high". If one does not have such previous experience, it is possible to start with a big value and determine the acceptance rate. For a good initial temperature this rate should be high. It depends on the actual problem, but an acceptance rate between 40% and 60% often is a good reference value.

Cooling scheme:

The probably most important aspect is the cooling scheme. More precisely, we have to find values for k and l and the value α. Usually α takes a value between 0.9 and 0.99. The value T can be decreased by $T := \alpha T$. But it can also be done in another way, for example by setting $T := \frac{T}{1+\beta T}$, where β is a constant close to 0.

Break condition:

We do not want to keep applying 2-Opt steps forever but rather want to terminate the algorithm eventually. For example, this could be done after a predefined number of iterations or if a sequence of s steps did not result in an improvement, with s being yet another parameter.

As concluding remark we want to mention that the 2-Opt step (or in general the k-Opt step) is just one possibility to generate from one round trip another round trip. In fact there are a lot more possibilities that can also be applied in the scheme of Simulated Annealing. For those alternative methods it is also necessary to determine the right strategies by performing experiments.