Methods and Heuristics for Learning and Optimization

Series 3: Simulated Annealing and Parallel Tempering for the Traveling Salesman Problem

Return no later than October 31, 2022 (13h00)

1 The Traveling Salesman Problem

The goal of this exercise is to implement the *Simulated Annealing* (SA) algorithm and use it to solve different instances of the *Traveling Salesman Problem* (TSP). In the TSP, a set of n cities is given and the distance between each of them is known (we assume there is a direct road between any two cities). The goal is to find the shortest tour that allows each city to be visited exactly once [1].

2 Simulated Annealing

We describe below the successive phases of the SA algorithm (figure 1).

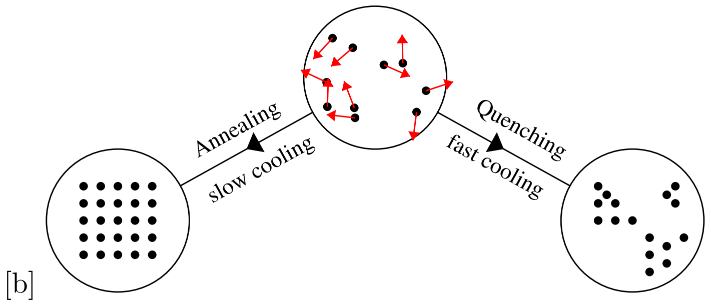


Figure 1: Representation of the physical process of annealing (figure from [1]).

1. **Initial configuration:** Start with a randomly generated path.
2. **Initial temperature:** Starting from the initial configuration, compute 100 transformations (permuting two cities) and compute the average energy change $\langle|\Delta E|\rangle$, where $\Delta E = E(x_{\text{neighbor}}) - E(x)$. The initial temperature then can be computed using the equation:
$$e^{\frac{-\langle|\Delta E|\rangle}{T_0}} = 0.5. \tag{1}$$
3. **Elementary Configuration Update:** Randomly transform the solution.
4. **Acceptance/Rejection rule:** Keep the update with the probability given by the Metropolis rule (figure 2):
$$P = \begin{cases} 1 & \text{if } \Delta E < 0 \\ e^{\frac{-\Delta E}{T}} & \text{otherwise} \end{cases} = \min(1, e^{\frac{-\Delta E}{T}}) \tag{2}$$

[1]Or, in more formal terms, to find a hamiltonian cycle of minimal length on a fully connected graph.

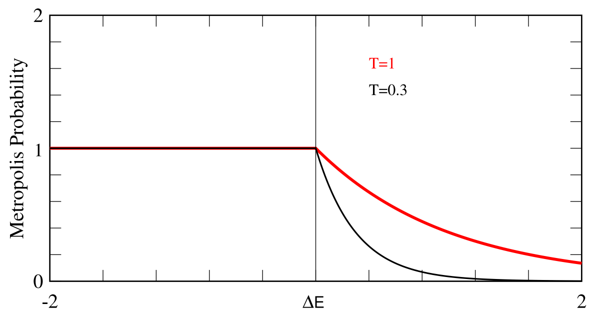


Figure 2: Metropolis rule of equation (2), taken from [1].

5. **Equilibrium conditions:** Consider that equilibrium is achieved if either $12n$ perturbations were accepted or $100n$ perturbations were attempted.
6. **Temperature Reduction:** Use the following simple rule:

$$T_{k+1} = 0.9T_k \tag{3}$$

7. **Freezing conditions:** The system is considered frozen if there was no fitness improvement during the last 3 temperature changes.

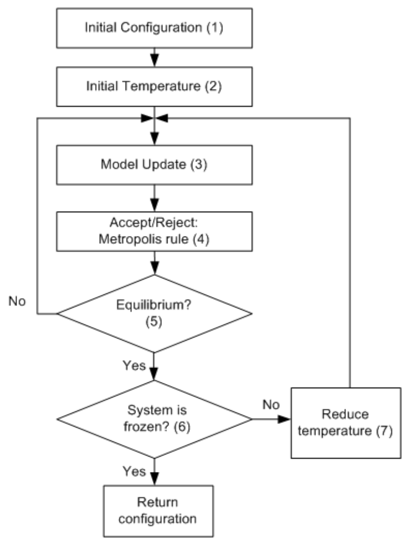


Figure 3: Simulated Annefloataling (SA) diagram.

The steps above are summarized in figure 3.

2.1 Configurations

For the TSP problem, a configuration is given by a sequence of cities; energy E of the configuration equals to the path length. Elementary configuration update amounts to randomly transposing 2 cities in the path.

3 Work to do

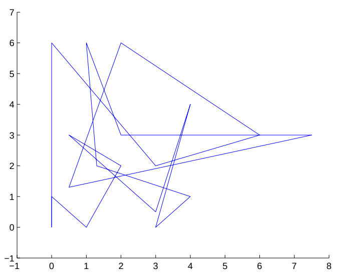
In all the experiments assume that you start and end your path in the first city, i.e. the first city might be a warehouse from where the goods are delivered to all the facilities (it means that $c_{n+1} = c_1$ must hold in (4).

You will find attached to this assignment two files that describe the city placements. Before starting using that files, start testing your algorithm in a simple test-case where all *facilities* are placed around a circle. In this case, you expect that your method finds a path that is (similar to) a circle. Run several times your SA algorithm on the two problems attached to this exercise and report the solutions as in (4).

Compare and discuss results of the SA method with those of a greedy algorithm [2], repeated 10 times.

For the two TSP problems, provide a visualization of the obtained optimal solutions by plotting the positions of the cities and the shortest paths obtained by the SA and the greedy algorithm.

The following picture displays an example of such a visualization:



Moreover, solve five randomly generated TSP problems of size 50, 60, 80 and 100. For these problems, report and comment on the means and standard deviations of the execution times and lengths of the paths of the SA and greedy algorithm (the latter averaged over 10 runs).

Test and discuss the impact of abrupt temperature reductions.

3.1 Practical Considerations

The implementation must comply with the following requirements:

- Your code should take one argument which is the name of a task definition file.
- The task definition file should have the same format as the provided files *cities.dat* and *cities*2*.dat*
- The program should solve the TSP problem and return an answer in the form of a sequence of cities corresponding to the shortest path and its length, i.e.:

$$[c_1, \ldots, c_n] : \sum_{i=1}^{n} d(c_i, c_{i+1}) \tag{4}$$

where $c_1, \ldots, c_n$ are cities and d is a distance measure.

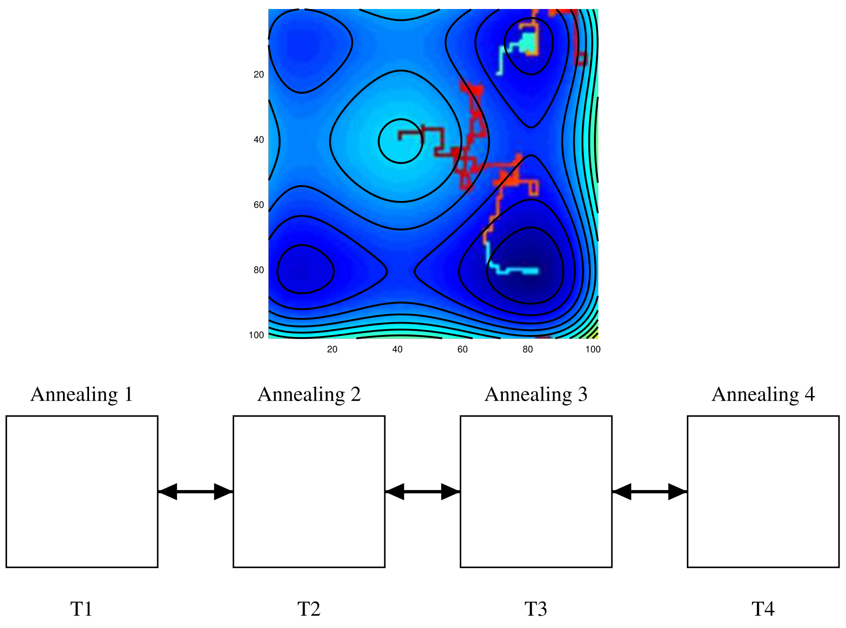


Figure 4: Graphs to interpret the parallel tempering systems. On the top, different PT systems trajectories are represented in the same figure representing the research space; the color represents the temperature of the trajectory (red high, blue low). On the bottom, the different system replicas are represented in association with the tempering temperatures.

4 BONUS (+1pt): Parallel Tempering

The PT algorithm (figure 4), also known as *replica exchange*, is an optimization method which can be seen as a parallel version of Simulated Annealing (SA). But where the temperatures are not reduced gradually. In fact, the analog physical process in this case is the quenching (figure 1). The PT method allows the replicas at different temperatures to swap complete configurations. As a result, replicas with low temperatures are able to escape local optimas and explore other parts of the search space.

More formally, consider M replicas of the SA methods, each at different temperature T_i. We assume $T_1 < T_2 < \ldots < T_M$. We consider a move where the replicas at T_i and T_j swap the configurations; the proposed swap is accepted with probability

$$p(i,j) = \min(1, e^{-(\frac{1}{T_j} - \frac{1}{T_i})(E_i - E_j)})$$

where E_i is the energy of configuration i. Intuitively, a swap is accepted with probability 1 if the replica with higher temperature has lower energy; if the replica with higher temperature has higher energy, then the probability of a swap depends on the differences of (inverse) temperatures and energies. Here we assume that swaps are only attempted between replicas with adjacent temperatures, i.e. $j = i + 1$ for $j = 1, \ldots, M - 1$. It is usually the state associated with the lowest temperature T_1 that is of interest since it will typically yield the estimate for the global optimum, while all others are required to generate other potential solutions.

An example of a sequence of swaps is presented in Figure 5.

[2]This algorithm simply requires to pick a city randomly, then move to the nearest one, then to the nearest one that has not been visited yet, etc

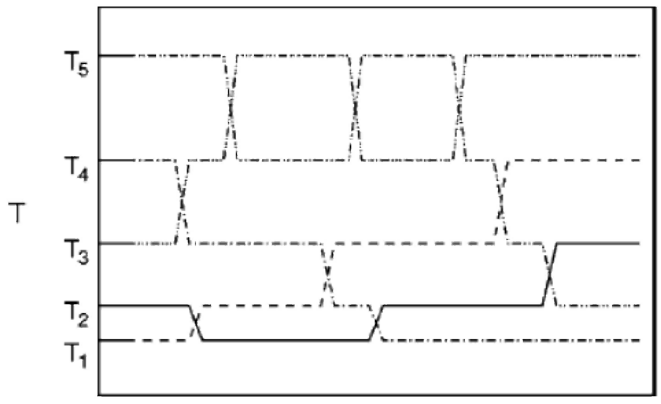


Figure 5: A sequence of swaps in Parallel Tempering.

4.1 PT for TSP

The main difficulty in applying PT is the choice of the temperatures $T_1, \ldots, T_M$. The temperature T_M should be chosen as in the case of the standard SA, whereas T_1 should be a small positive value. Different options are possible for choosing temperatures $T_2, \ldots, T_{M-1}$. In particular, temperatures could be distributed geometrically or arithmetically. It is up to you to decide the good setup.

The PT algorithm works as follows: (i) it individually updates the states in each replica, and then (ii) it attempts a swap between a random pair of consecutive replicas. It continues with (i) if the maximum number of iteration, t_{max}, is not exceeded (it is up to you to select the meaningful value of t_{max}).

Concerning the number of replicas try $M = \{2, 5, 20\}$.

5 Report

Each student is required to give back a *personal* work consisting of a code and a concise but precise report in PDF format (4-5 pages) displaying an introduction of the problem to be solved, a description of employed metaheuristics, experiments carried through with corresponding results, and a discussion. Both report and code (*Surname Name TP number*) have to be uploaded on Moodle (TP3).

References

[1] B. Chopard and M. Tomassini, *An Introduction to Metaheuristics for Optimization.* Cham: Springer International Publishing, 2018.