# physics760 - Computational Physics

**Monte Carlo Optimisation: The traveling salesman problem**

Folkert Bismark
s6fobism@uni-bonn.de

Nicolas Boeing
nicolas.boeing@gmail.com

15. march 2018

# Contents

# 1 Introduction

The traveling salesman problem (TSP) envisions a salesman willing to sell his products in multiple cities. In order to visit each city once as efficiently as possible, his path will need to be optimized accounting for the distance between each city and the speed of the saleman.
In the real world this problem typically occurs for parcel delivery companies delivering parcels to many people using as little total pathway as possible.
Another question of interest in modern physics is how to define the wiring routes, considering the design rules, on a PCB to reduce wiring length, number of drill holes and the position of active electronic modules, e.g. transistors or diodes [1].
With growing number of cities to be visited the complexity of the TSP grows with the factorial of number of cities and is therefor impossible to brute force with a large amount of cities. However, algorithms can be used to approximate the most efficient solution.

# 2 Theory

In the first part of our discussion we are going to consider a random configuration of cities, as an example in figure 2.1. As a following consequence one has to assign each station a number when it should be passed by the salesman. The order of stations with the smallest total length $L_{travel}$ should be reached.
In order to evaluate the minimum of $L_{travel}$ one has to consider all possible permutations of the stations and determine $L_{travel}$. Instead of considering each permutation and evaluating $L_{travel}$, one makes a thermodynamical approach.

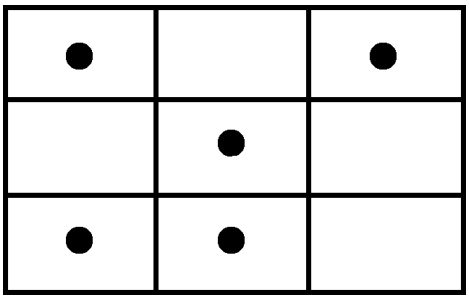

Figure 2.1: The figure shows a lattice with multiple circles, representing a city or station of the traveling salesman.

The idea of this approach is to describe the properties of the 'TSP'-system by a thermodynamical spin system with two states. Therefore each state of the spin system corresponds to one possible configuration of the stations (cf. 2.2). This kind of approach means that the energy of the spin system is the 'characteristic length' which correspond to the 'characteristic length' of the TSP-system being the total path the salesman has to travel.
In this sense one has to minimize the total energy of the corresponding spin system. The total energy of a spin system being in a particular state is given by equation 2.1.

$$E = A \cdot (n_+ - n_-) \tag{2.1}$$

where $n_+$ is the number of spins being in the 'up' state, $n_-$ is the number of spins being in the 'down' state determined by the quantization axis and $A$ is a coupling

constant.

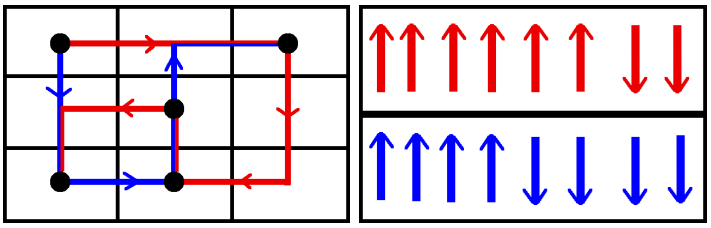

Figure 2.2: In the left figure is shown two possible pathes of the salesman to get to every station. In the right picture is the corresponding spin orientation describing equivalently the TSP-system.

An example of the used approach is shown in figure 2.2, where we have assumed that the velocity of the salesman is in each direction the same. We have also assumed that he can only walk to the neighbouring cell, which does not enclose an angle to the x- and y-axis. The figure on the left side is showing two possible ways, where the red curve is longer than the blue curve. Consequently the two spin-system, representing the two TSP systems, shown in the right figure have to have the same properties. Therefore the energy of the blue spin system is lower than the energy of the red spin system. It can be seen by comparing the number of spins showing up and the spins showing down, which lead to different energies according to equation 2.1. In order to name which way is the shortest way, one has to determine the Boltzmann probability and evaluate the corresponding mean energy $\bar{E}$ for a given state. This can be done by computing equation 2.2 and 2.3.

$$P_{state} = exp(-E_{state}/T) \tag{2.2}$$

where $T$ the temperature and $E_{state}$ is the total energy of the system being in some particular state. The state is defined by the configuration of the spin system, which equals a permutation of the stations for the traveling salesman.

$$\bar{E} = \left[\sum_{state} E_{state} \cdot P_{state}\right] / \left[\sum_{state} P_{state}\right] \tag{2.3}$$

# 3 Algorithm

In the following chapter the algorithm used to implement the thermodynamic approach outlined above is described. It is taken from [2] and is based on the well knwon Metropolis-Hastings algorithm.
The basic idea of the algorithm is to compute the distances between $N$ stations, stored in a $N \times N$-matrix and to find a permutation of the stations $c_{i,\dots,N}$ such that the total distance, calculated via equation 3.1, becomes minimal:

$$d_{total} = d(c_1, c_N) + \sum_{l=1}^{N-1} d(c_l, c_{l+1}) \tag{3.1}$$

where $d(c_i, c_j)$ describes the distance of the $i$-th to the $j$-th station.
In order to achieve this the following steps are executed:

0. Choose a random starting permutation $s_i$, i.e. connect all cities in a random order. Additionally chose a temperature T.
1. Set $c_k = s_k$ and calculate the length of the path according to eq. 3.1
2. Set $i = 1$.
3. Generate a random integer $j$ such that $1 \leq j \leq N$, $j \neq i$.
4. Construct a new permutation $t_k = c_k$ while flipping the path between $i$ and $j$: $t_{\tilde{i}+k} = c_{\tilde{j}-k}$ for $k = 0, \dots, \tilde{j} - \tilde{i}$, $\tilde{i} = \min(i, j)$, $\tilde{j} = \max(i, j)$.
5. Calculate the length of this new permutation $d'$.
6. If $d' < d$, jump to step 7, otherwise generate a new random number $0 < x < 1$. If $x < \exp((d - d')/T)$ jump to step 7, otherwise jump to step 8.
7. Accept the trial permutation $c_k = t_k$ and set $d' = d$.
8. Increment $i$ by one, and if $i \leq N$ go back to step 3, otherwise go to step 2.

The amount of iterations to be done with this algorithm is set by hand and depends on its performance with the configuration at hand. The steps of the algorithm are

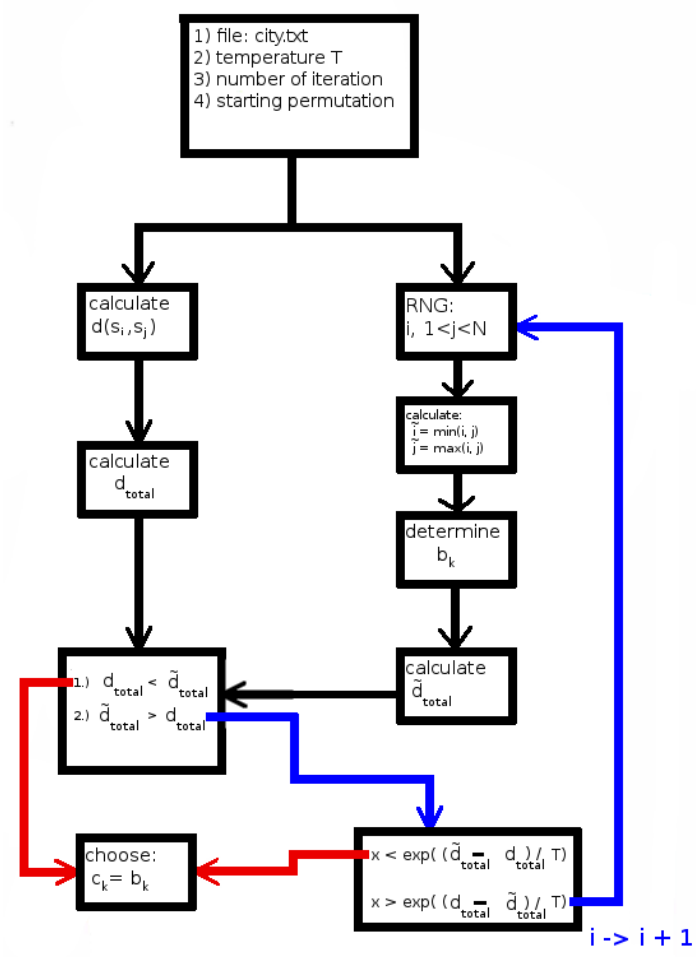


Figure 3.1: The figure shows schematically a block diagram of the important steps of the algorithm.

also ôutlined in fig. 3.1.

The temperature used in the acceptance rate calculation has to be set by hand and can either be determined with an educated guess or optimized, the same is true for the number of iterations.
This algorithm can also be further improved. Firstly the temperature can be reduced each time an equilibrium is reached, getting closer to the optimal value each time. Also, for the region close to the optimal value $j$ can be calculated differently in order to favor local flips and increasing the acceptance rate (Step 3') This is done by generating $j$ such that $1 \leq m \leq n \leq N/2$ with $n$ fixed (typically $N/10$) and $m = min|j-i|, |j-i+N|, |j-i-N|$.

The implementation of this algorithm is done using a C++ program. City locations are read in from `cities.txt` which can be generated automatically by using the helper script `CreateCities.py`. A config file `config.txt` is used to configure variables such as the temperature, number of iterations and if the temperature should be automatically dropped after every 1/4 of iterations. Output files are `log.txt` for the distance after each iteration and `path.txt` for the path taken in the final iteration.

# 4 Analysis

At the beginning of our investigation we have to confirm that the algorithm behaves like expected. In order to do this, the algorithm runs a number of iteration, e.g. 20000, shown in figure 3.1. Aditionally the temperature is changed in the sequence $T = (0.1, 0.01, 0.001)$ and the total distance $d_{total}$ is plotted against the number of iterations $N_{iteration}$. In the paper (cf. [2]) a result is given for two cases, in one case the cities are oriented like a circle. In the second example the cities are distributed on a rectangular grid.
The true minimum for the circle is $d_{total} = 6.28$ for the unit circle, where 100 cities are placed. It can be achieved by the used algorithm, figure 4.1 shows a comparison of our result and of the given result [2].

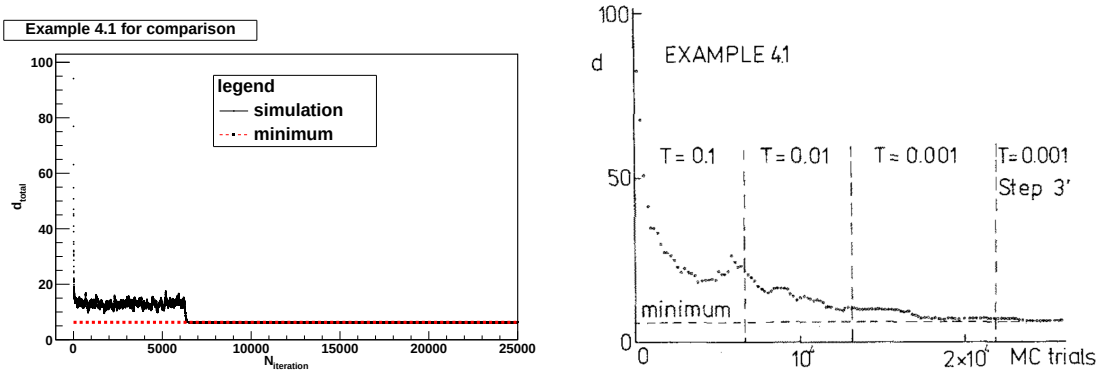


Figure 4.1: The left figure shows the result the used algorithm. For comparison, on the right, an example from [2], page 7.

The comparison in figure 4.1 shows that the true minimum of $d_{total} = 6.28$ can be reached in both cases. It can be observed that the number of used iterations is in our case $N_{iteration,min} = 6382$ and in their case $N_{iteration,min,lit} = 25000$.
The determination of the $N_{iteration,min}$ has been done by performing a linear fit of the form $d_{total} = a \cdot N_{iteration} + b$ on the change to the minimum. The fit and its parameters are shown in figure 4.2. In order to estimate $N_{iteration,min}$ one has used a value for $d_{total} = 6.28$ being the true minimum.

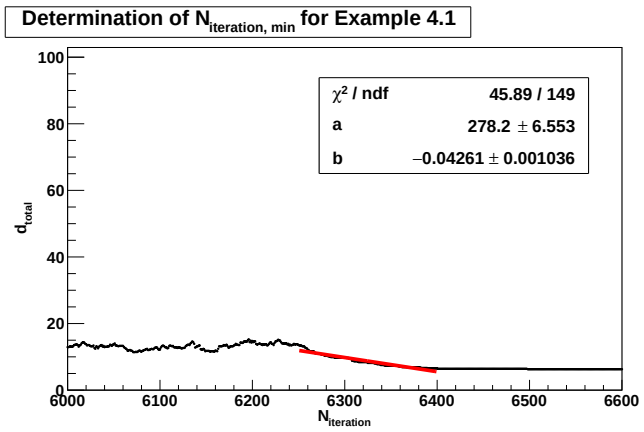


Figure 4.2: The graph is showing the used fit and its function.

In order to determine which approach is more useful to find the true minimum, we have also investigated the TSP on a $10 \times 10$-grid with 100 cities distributed on the grid. In the given literature (cf. [2], p. 6, Example 4.3) the true minimum should be at $d_{total} = 100$, but could not be reached with the proposed algorithm. In our case it has been achieved at $N_{iteration} \approx 8700$, see fig.4.3.

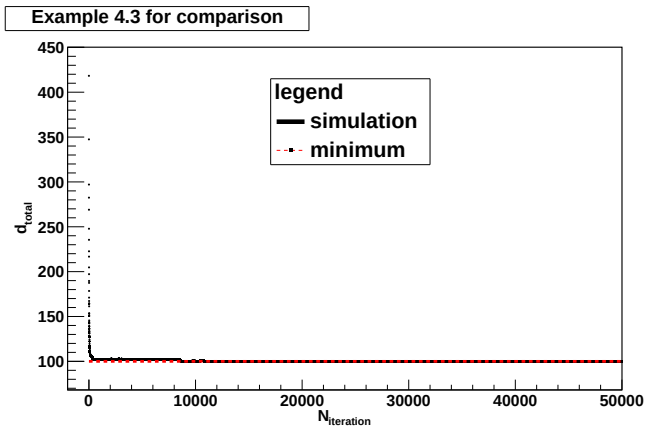


Figure 4.3: The graph is showing our result

The use of a python script ('PathPlotter.py') allows to draw the latest path of the 100 cities distributed on a grid, it is shown in figure 4.4.

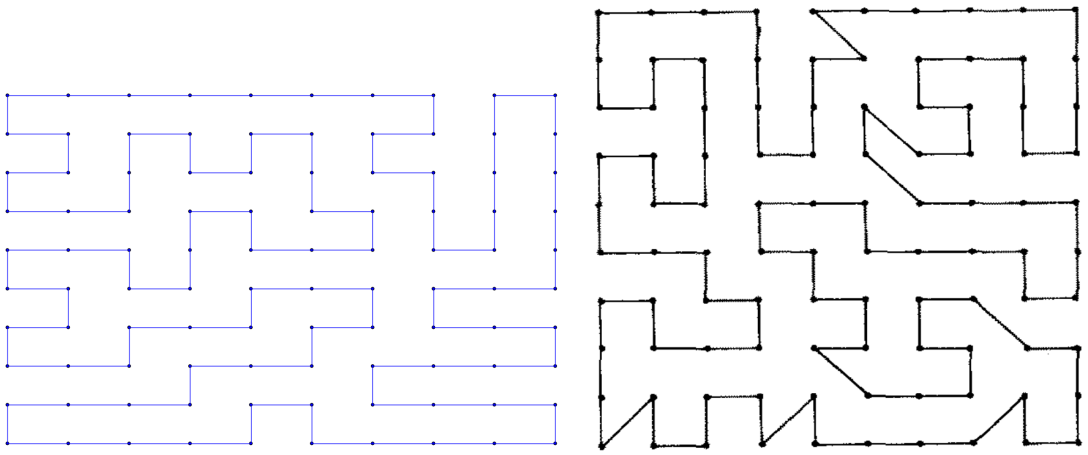

Figure 4.4: The left figure is showing the path on the lattice with 100 cities by using step 3' at the beginning of the algorithm. In comparison is shown in the right figure the path of the proposed algorithm (cf. [2], p. 7, Example 4.3).

Additionally we have investigated the dependence of the computing time on the number of cities, shown in figure 4.5.

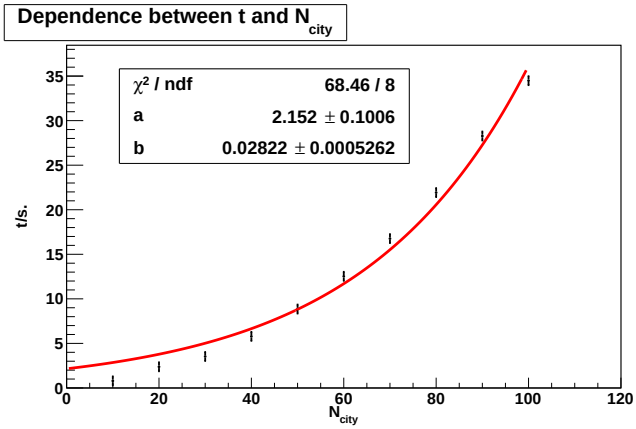


Figure 4.5: The graph is showing the change of the computing time for different number of cities and for two different configurations.

The graph 4.5 shows that the computing time does depend exponentially on the number of cities $N_{city}$. In this case one has used $N_{iteration} = 12000$ for a temperature of $T = 0.1$ changed three times by a factor of $1/10$. The functional behaviour can be modeled by $t = a \cdot exp(N_{iteration} \cdot b)$ , with $a$ and $b$ being fitting parameters, the corresponding error on $t$ has been setted to $\Delta t = 0.5$s, because this error has occurred on each simulation point when the algorithm has been compiled several times.

Another question of interest is whether the algorithm is able to find the permutation for the shortest distance $d_{total,min}$. It has been tested by considering two $10 \times 10$-grids, called 'islands', and a distance of five length units. The final path found by the algorithm is shown in figure 4.6.

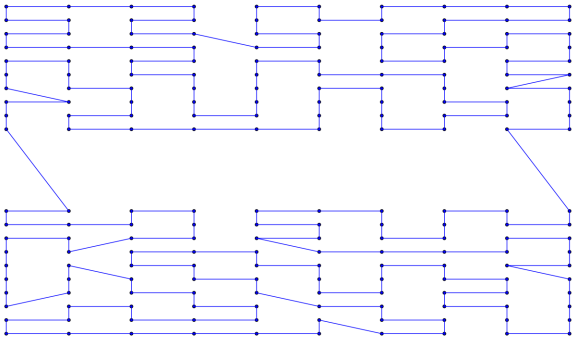

Figure 4.6: The figure is showing the path for two islands being $10 \times 10$-grids.

In the figure can be seen that the algorithm does find the permutation with the shortest distance, because the cities lying within an 'island', are connected first. The algorithm is able to find a connection to the second 'island' and passes all cities in this 'island'. Afterwards it creates a connection to the first 'island', cf. 4.6.
In the last part of the analysis we have make a change of step 4 to step 4', where the position of the $i$-th and $j$- th city has been changed. The difference between these two steps is shown in figure 4.7.

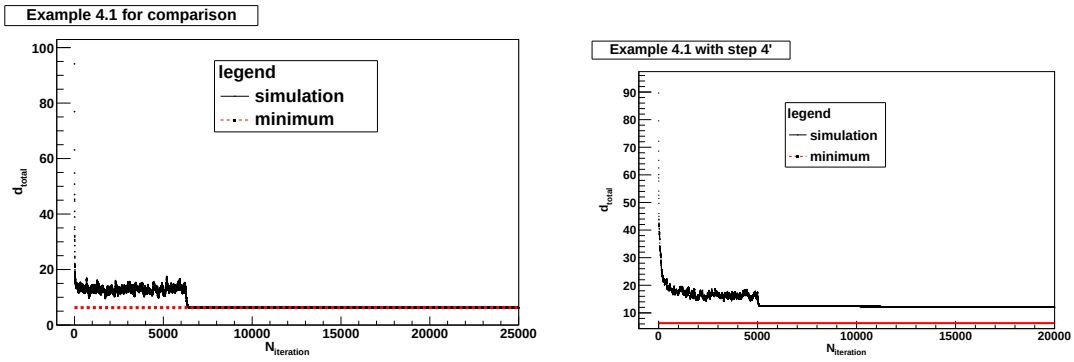


Figure 4.7: The left figure is showing the behaviour for step 4. In the right figure shows for comparison step 4'.

The comparison of these two cases (cf. 4.7) shows that when one only changes the position (step 4') and does flip the distance between the $i$-th and $j$- th city the true minimum can not be reached by the algorithm.

# 5 Conclusion

The proposed thermodynamic algorithm does indeed approximate a solution to the TSP within a reasonable amount of computation time. The optimization of step 3, 3', which is being proposed in the paper, improves the algorithm further. The results in example 4.1 and 4.3 have shown, that the true minimum can be achieved in both cases. In the case of a circle $d_{total,min} = 6.28$ and for a square (lattice) $d_{total,min} = 100$. The comparison of the number of iteration between the literature and our results shows an improvement, because the equilibrium state has been achieved for a lower number of iterations (cf. 4).
Another problem of interest being proposed in the used literature ( [2]) is the 'island' problem, where two squares are apart from each other. The optimizied algorithm does solve the problem, therefore it is able to find the permutation with the smallest distance.

# Bibliography

[1] Rajesh Matai, Surya Singh and Murari Lal Mittal (2010). *Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches, Traveling Salesman Problem, Theory and Applications*, Prof. Donald Davendra (Ed.), InTech, DOI: 10.5772/12909.

[2] V. Cerny, *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*, JOTA, Vol. 45, No 1, 1985