Traveling Salesman Problem
project paper

Karl Franzens University of Graz
Monte Carlo Methods

by Prof. Dr. Denes Sexty

Maximilian Gschaider BSc
gschaider@student.tugraz.at

01.04.2024

Contents

1 Introduction

Monte Carlo Methods play a powerful method in different areas of science, where randomness meets computation to solve complex problems. For the course *Monte Carlo Methods* held by *Prof. Dr. Denes Sexty* at the *Karl Franzens University of Graz* in the semester *23/24* it was wished to work out one of the handed out problems.
As the Traveling Salesman Problem (TSP) is widely known and historically one of the first problems in the field of combinatorial optimization that has intrigued mathematicians, computer scientists, and practitioners for centuries, I decided to go with it.
There are few different possible methods to solve it with certain accuracies, calculation times and computational complexities. In this paper I'll focus on the solution via the Metropolis Hasting algorithm, which was discussed in the corresponding lecture.

The essence of the TSP lies in its quest to find the shortest possible route for a salesman to visit a set of cities exactly once and return to the starting point.

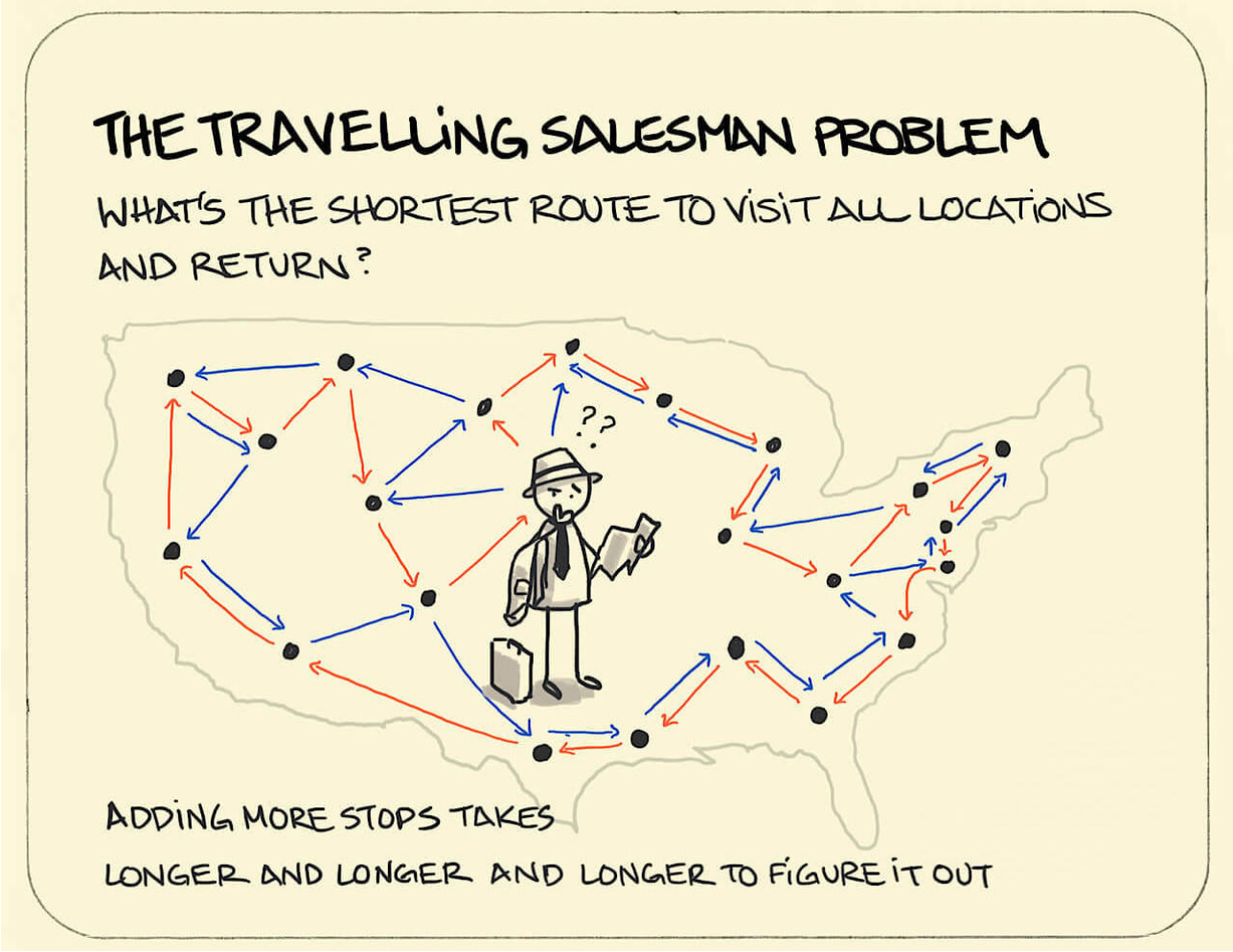

Figure 1 Classical Travelling Salesman Problem (TSP) [1]

2 Methodology

For solving the Traveling Salesman Problem (TSP) mathematically it first has to be transformed into a suitable model. This can be done by the help of graph theory by vertices (nodes) and lines (edges). The cities in the plane can be represented therefore by nodes and the paths

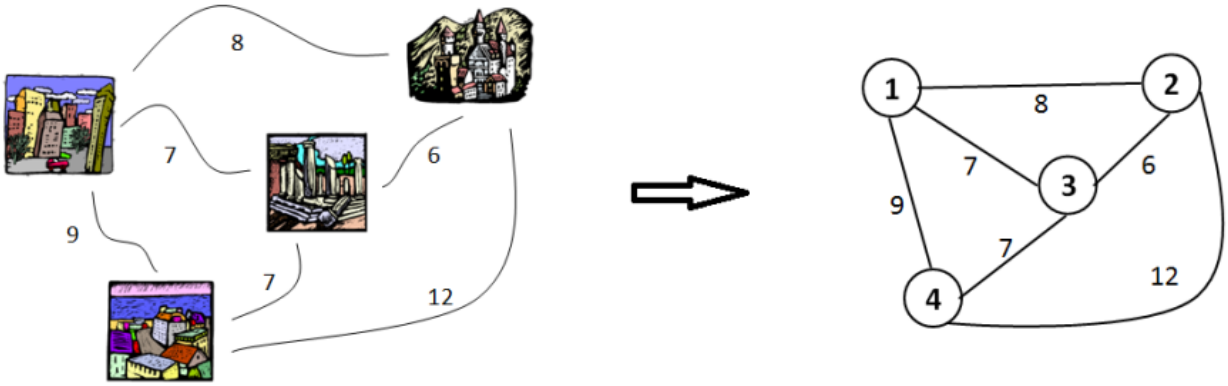

Figure 2 Travelling Salesman Problem (TSP) represented in Graph Theory [4]

between each city consequently by lines, whereby a direction is added. For each individual city pair (i, j) there exists a path $c_{ij} \geq 0$ which is simple the total distance between those cities for the symmetric TSP, which means that the traveling cost is direction independent (going from i to j results in the same duration that from j to i / = directed graph). In addition in most cases the graph is said to be completed, so between two nodes there is always one line.
A symmetric TSP is also metric, when its line lengths fulfill the triangle inequality equation. This means that detours are not worthwhile because the direct connection from i to j is never longer than the path from i to j via a third node k:

$$d_{ij} \leq d_{ik} + d_{kj} \tag{1}$$

When choosing a plane in the real space this distance can be calculated by using the euclidean distance metric:

$$d(i, j) = \sqrt{(x_i - y_i)^2 + (x_j - y_j)^2} \tag{2}$$

Whereby x and y correspond to the individual projection of the city onto the orthonormal basis vectors $\hat{e}_x$ and $\hat{e}_y$ of the two dimensional system.
The corresponding value $d(i, j)$ induced by the metric can be thus because of its scalar nature interpreted as an associated cost of the event traveling from city i to city j. In other words, the time t which the salesman traveles between two cities (i, j) with distance d with a uniform speed v is given by: $t = \frac{d}{v}$, which represents indeed an associated cost of the system.
When dealing now with multiple cities N, where $N > 3$ for a non-trivial path sketching, there are multiple ways to arrange those paths given by the corresponding value: $(n-1)!/2$ for a symmetric TSP. One can use the assumption about the induced associated cost to extract information of the modeling behaviour of the traveling process between multiple path combinations. Ultimately we want to choose the overall shortest path (lowest energy) between all N cities. This is now equivalent to choose the total cost which is minimized under the constraint that each city has to be visited once.

3 Theoretical background for computational implementation

As already stated we're dealing with a combinatorial optimization problem where we want to get the shortest path between all N cities on a two dimensional plane, where each path (distance) is given by the euclidean metric $d(i,j)$. We can assign those distances to a matrix $\hat{d}$ with entries d_{ij}, where because of symmetry of the problem (symmetric TSP) the matrix is also symmetric. Additionally the trace of the matrix is zero $\mathrm{tr}(\hat{d}) = 0$, because each individual node has no self-path (distance from city k to itself must be zero). The entries d_{ij} represent the individual distances from city i to j or consequently from j to i: $d_{ij} = d_{ji}$.
The goal is now to find a route for a round trip represented by the permutation $p(i)$ of $1, .., N$ such that the total distance travelled

$$D = \sum_{i=1}^{N} d_{p(i),p(i+1)} \tag{3}$$

is minimal. In addition we define $p(N+1) = p(1)$ as boundary condition for a round trip [6]. For solving the problem it is stated that one should choose the Metropolis Hasting algorithm with simulated annealing to find a route with "minimum annealing". Where the term *simulated annealing* comes from the idea to simulate the cooling process for example the "annealing process" in metallurgy. But firstly start with the Metropolis algorithm and its properties.

3.1 Markov Chain Monte Carlo methods

The Metropolis algorithm is a Markov-Chain-Monte-Carlo (MCMC) method for creating states of a system corresponding to the Boltzmann distribution. The new state ϕ_{i+1} is only dependent of the previous state ϕ_i.

$$\phi_i \to \phi_{i+1} \to \phi_{i+2} \to ...$$

MCMC is an efficient approach to perform sampling in system with high dimensions, whereby the probability density function (PDF) $\pi(x)$ is often dominated by only a small part of the space [3]. The configuration of the states $\phi(x)$ are created iteratively in such a way that their distribution are corresponds to a desired distribution.
The probability for going from a state ϕ to a state ϕ' is given by the transition probability $T(\phi'|\phi)$, whereby the transition probability T does not depend on the iteration step i in the chain but only on ϕ, which is a characteristic property of the Markov process. If one accumulates all transition probabilities of states ϕ' the sum should result to one because of normalization.

$$\textstyle\sum_{\phi'} T(\phi'|\phi) = 1$$

One can now write out the balance equation for equalibrium for describing the behavior of a system similar to the equation of continuity.

$$\pi(\phi) = \textstyle\sum_{\phi'} T(\phi|\phi')\pi(\phi') = \underbrace{\sum_{\phi'} T(\phi'|\phi)}_{\mathbb{1}} \pi(\phi)$$

When we choose $T(\phi|\phi')$ such that we have detailed balance, which means that the current is the same in both directions (input / output).

$$T(\phi'|\phi)\pi(\phi) = T(\phi|\phi')\pi(\phi')$$

which can be also written in matrix notation: $T_{ij}\pi_j = T_{ji}\pi_i$.

3.2 Metropolis algorithm

We now can use the MCMC method to describe the Metropolis algorithm in its general form.

Algorithm 1: Metropolis generic algorithm

We start from a state ϕ_0 and after $n-1$ steps do:
propose $T_0(\phi'|\phi_{n-1})$
define acceptance probability $T_A(\phi'|\phi_{n-1})$
if *accepted through acceptance probability* $T_A(\cdot)$ **then**
  $\phi_n = \phi'$
else
  $\phi_n = \phi_{n-1}$
end

If additionally the detailed balance should be satisfied, the following expression is valid:

$$T_0(\phi|\phi')T_A(\phi|\phi')\pi(\phi') = T_0(\phi'|\phi)T_A(\phi'|\phi)\pi(\phi)$$

Detailed balance is a specific property in Markov chains, which means that is not recognizable if the process is forward directed or backward.
This can be rearranged to the ratio of the individual probability density functions (PDF) $\pi(\cdot)$ of the states.

$$\frac{T_0(\phi|\phi')T_A(\phi|\phi')}{T_0(\phi'|\phi)T_A(\phi'|\phi)} = \frac{\pi(\phi)}{\pi(\phi')}$$

Usually one will choose the assumption of irreversibility of proposal, which just means $T_0(\phi|\phi') = T_0(\phi'|\phi)$. For the PDF it is convenient that one uses the Boltzmann distribution, which describes the occupation probability of particles in physical system (non-quantum mechanically treated), which can be expressed through the action $S[\phi]$: $\pi(\phi) = e^{-S[\phi]}$. Plugging this into the previous equation for both states ϕ and ϕ' the ratio results in the difference of actions:

$$\frac{\pi(\phi)}{\pi(\phi')} = e^{-(S[\phi]-S[\phi'])} = e^{-\Delta S}$$

In this case the acceptance probability $T_A(\cdot)$ results to:

$$T_A(\cdot) = \begin{cases} e^{-\Delta S} & \text{if } \Delta S > 0 \\ 1 & \text{if } \Delta S \leq 0 \end{cases} \tag{4}$$

This process will optimize the transition probability $T_0(\phi|\phi')$ in the following way: If the change ΔS is small the configuration space exploration is slow, but the new state is indeed energetically equivalent or more favorable, we accept it with $T_A(\cdot) = 1$. If the change ΔS is big the acceptance is small and the new state energetically unfavorable. This state will only be accepted with the probability p_A (5), where a random number q will be sampled from a uniform distribution and if q is smaller than p_A, the new state will be accepted otherwise it will be rejected.

$$p_A(\cdot) = \min[1, \exp(-\beta\Delta E)] \tag{5}$$

We can now use those assumptions about state configurations and the goal to retrieve a more favorable energy induced by the state configurations of the system to reduce the total energy of the system by iteration. Due to the fact that the total energy may not only have one minima in its objective function this process must not converge ultimately in a feasible amount of iterations to its global minima.

4 Simulated Annealing

To gain additional information for finding the global minima, the Metropolis algorithm will be used with Simulated Annealing. Simulated Annealing is a heuristic approximation method, which can be used to find a (approximate) solution of optimization problems, which because of complexity can not be explored in full depth.
The idea behind it comes from cooling processes (annealing) in metallurgy. After a metal is heated up the *slow cooling process* enable the atoms to align in patterns and build stable structures. Consequently a low energy state near a optimum will be targeted.
Compared to the Metropolis Algorithm it is characterized by the fact that the temperature T will be decreased during iterations or consequently the inverse temperature β will be increased. This enables the escape of local optima to find a better one. To find a more favorable state of a system, which can be described by the Boltzmann statistic, one case use the corresponding probability distribution for a microstate with energy $E_i \geq 0$:

$$p(E_i) \propto \exp(-\beta E_i) \tag{6}$$

The energy of the energetically most favorable state is notated as E_0. The probability converts now consequently to the following because of an independent factor of E_i:

$$p(E_i) \propto \exp(-\beta(E_i - E_0)) \tag{7}$$

Because E_0 is the energetically more favorable state, the following relationship must hold for $\beta > 0$:

$$E_i - E_0 \geq 0 \tag{8}$$

Consequently the exponent is strict negative and with decreasing temperature T (increasing β) the exponent gets larger, which means the probability decreases for finding an excited energy state with a minimum value of E_i. So by decreasing the temperature T of the system slowly (cooling process), the energetically favorable state will be found with higher probability.

5 Code Implementation

With the stated theoretical framework we can now implement the Traveling Salesman Problem with the Metropolis algorithm with Simulated Annealing. For illustration the rough project structure with its algorithms will be shortly sketched [2] (the full code is in [2] available).

Algorithm 2: TSP with Metropolis algorithm and Simulated Annealing

def City Map(N : int): $\rightarrow$ array; Initializing a city map with N cities on a unit square with random numbers sampled from a uniform distribution
def Metric(i : dupel, j : dupel): $\rightarrow$ float; Calculating the euclidean distance between two cities i and j with metric (2)
def Matrix assignment(cities : array): $\rightarrow$ array; Assigning all N cities to a symmetric matrix
def Total distance(tour : list, d : array): $\rightarrow$ float; Calculates the total distance of a tour through the indices provided by the list tour
def Tour swapping(tour : list): $\rightarrow$ list; Swapping the initial tour by changing two indices of the tour which are randomly chosen (range of the initial list)
---
def Metropolis(cities : array, M : int, N_s : int $\rightarrow$ best tour, best energy; Implements the Metropolis algorithm with with acceptance probabilities (4, 5) and Simulated Annealing (7)
calculating increment $\Delta\beta$ based on β_{min}, β_{max} and M
initialization of β through β_{min} (Simulated Annealing)
initialization of first tour and energy $\rightarrow$ best tour and energy
for *m in M* **do**
    calculation of proposed tour [*Tour swapping()* and *Total distance()*]
    for *n in* N_s **do**
        calculate Boltzmann factor $B_f(\beta, \Delta E)(7)$
        calculate energy difference ΔE (8)
        sample q from uniform distribution
        if $\Delta E < 0$ *or* $q < B_f(\cdot)$ **then**
            current tour = proposed tour
            current energy = proposed energy
        end
        if *current energy* $<$ *best energy* **then**
            best tour = current tour
            best energy = current energy
        end
    end
    increase β by increment $\Delta\beta$
end
return best tour, best energy

The initial functions are defined which built the base for the main function *Metropolis(·)* where the actual calculation begins. In two for loops the best tour and energy are determined within the iteration schematic whereby the inverse temperature β is increased by its increment $\Delta\beta$ after N_s sweeps. Depending on the two iteration counters M (Simulated Annealing sweeps) and N_s multiple possible optima for the best energy and tour can be returned by the Optimization algorithm.

6 Results and discussion

The sketched algorithm [2] will be implemented in a *Python* module. Firstly the algorithm is tested on functionality on a simple tour with $N = 10$ cities and a random seed (seed $s = 0$ from Numpy's random.seed module [5]) for reproducibility. Firstly we have to define plausible values for β_{min} and β_{max} for calculating the increment $\Delta\beta$. Because we deal with physical systems and limited computational power one can determine the right values by limiting the temperature to a physical meaningful range based on the Boltzmann constant k_B. With this assumption and the formula for the inverse temperature $\beta = (k_B \cdot T)^{-1}$ the range is set to: $\beta_{min} = 10\text{e+}10$ and $\beta_{max} = 10\text{e+}100$. The increment $\Delta\beta$ results now to the following:

$$\Delta\beta = \frac{\beta_{max} - \beta_{min}}{M} \tag{9}$$

Let's first test the framework with $N = 10$ cities and $M = 100$ iteration and $N_s = 100$ annealing sweeps. The plot is visible in figure [3], whereby the initial tour based on the randomly sampled cities and the best tour is plotted with the order in which the respective tour is completed. The

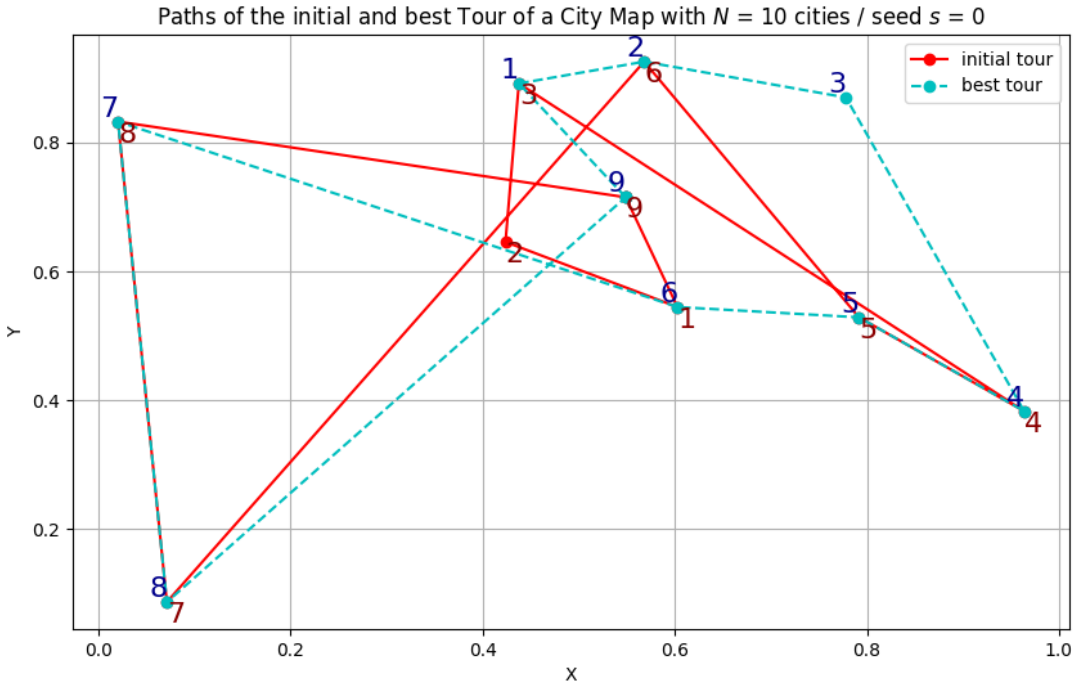

Figure 3 Comparison plot of the path of initial and best tour with $N = 10$ cities with $M = 100$, $N_s = 10$ and resulting $\Delta\beta$ (9) / random seed $s = 0$ [5]

depicted tour in red (with the corresponding numeration in dark-red) is the initial tour which is drawn by the random sampled cities in the unit map. We can already tell from a closer look that the drawn tour will not be the fastest (lowest cost) one. When comparing the tour with the best found in the optimization process one can see that the best tour which is found during the optimization process of the objective function has an indeed lower energy ϵ than the initial tour.

Table 1 Comparison between initial and best tour regarding their tour sequence and energy ϵ [3] with $N = 10$ cities with parameters $M = 100$, $N_s = 10$ and resulting $\Delta\beta$ (9) / random seed $s = 0$

tour	tour sequence	energy ϵ / 1
initial	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	4,8
best	[0, 3, 6, 9, 4, 5, 1, 8, 7, 2]	3,7

What if we increase the number of iterations of M and N_s further? In the following plot this is done by setting now M to 1000 and N_s to 100. We can see that the sequence of the

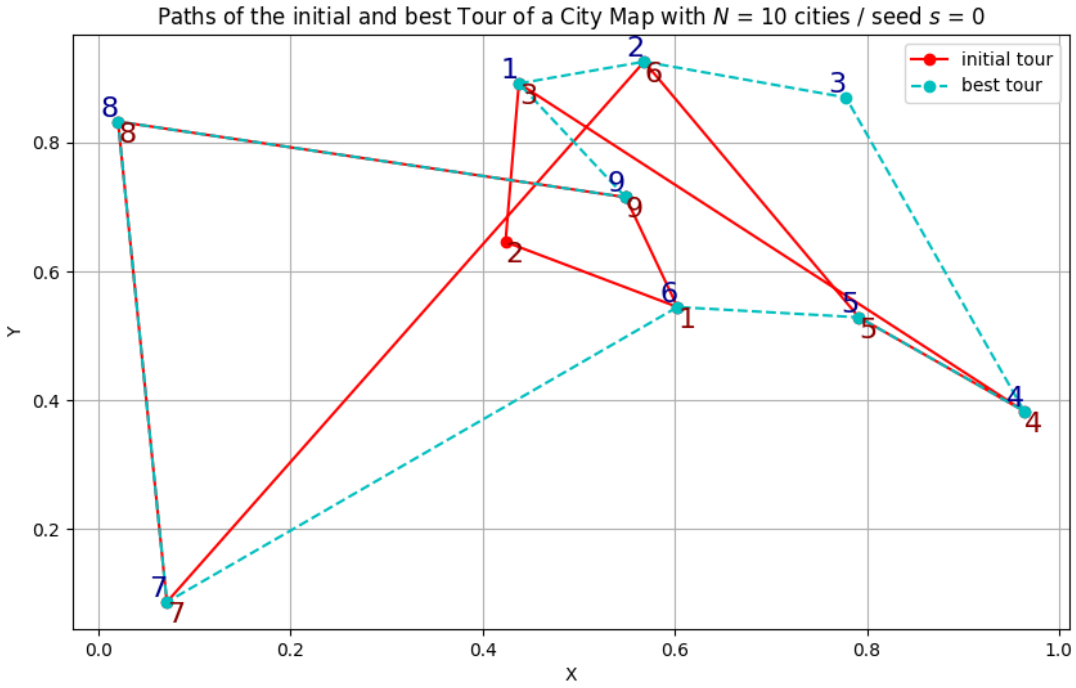

Figure 4 Comparison plot of the path of initial and best tour with $N = 10$ cities with $M = 1000$, $N_s = 100$ and resulting $\Delta\beta$ (9) / random seed $s = 0$ [5]

first 6 paths are exactly the same as in figure [3] but the path from 6 to 7 and 7 to 8 now has changed. Because the total distance is shorter the energy is consequently lower. But we've to

Table 2 Comparison between initial and best tour regarding their tour sequence and energy [4] with $N = 10$ cities with parameters $M = 1000$, $N_s = 100$ and $\Delta\beta$ (9) / random seed $s = 0$

tour	tour sequence	energy ϵ / 1
initial	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	4,8
best	[0, 3, 6, 9, 4, 5, 1, 7, 8, 2]	3,5

keep in mind that is not always the case that after increasing M or N_s a lower energy is definitely found which is illustrated in the figure [5 / 6]. The lowest energy (total tour distance) is the global minimum of the objective function of different path combinations, which is possibly found by sampling the energies through the Metropolis algorithm with Simulated Annealing. By increasing M the increment $\Delta\beta$ will be decreased and therefor the exploration in β increase, which increases the likelihood for finding the overall minima.

Let us now shortly analyze the influence of the two numerator parameters M and N_s on finding the best energy ϵ on each optimization round.

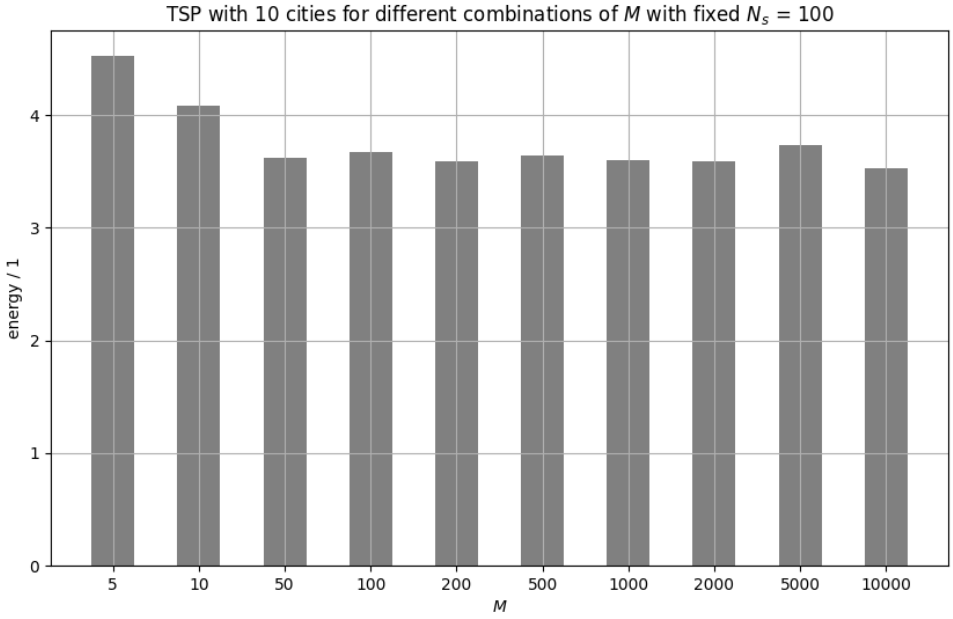

Figure 5 Energy values for different combinations of M for fixed $N_s = 100$ with seed $s = 0$

The average energy $\bar{\epsilon}$ and the variance $\nu(\cdot)$ in the first case where $N_s = 100$ is fixed and different values for M are tested results to: $\bar{\epsilon} = 3{,}42$ and $\nu(\cdot) = 0{,}16$ [Fig. 5].

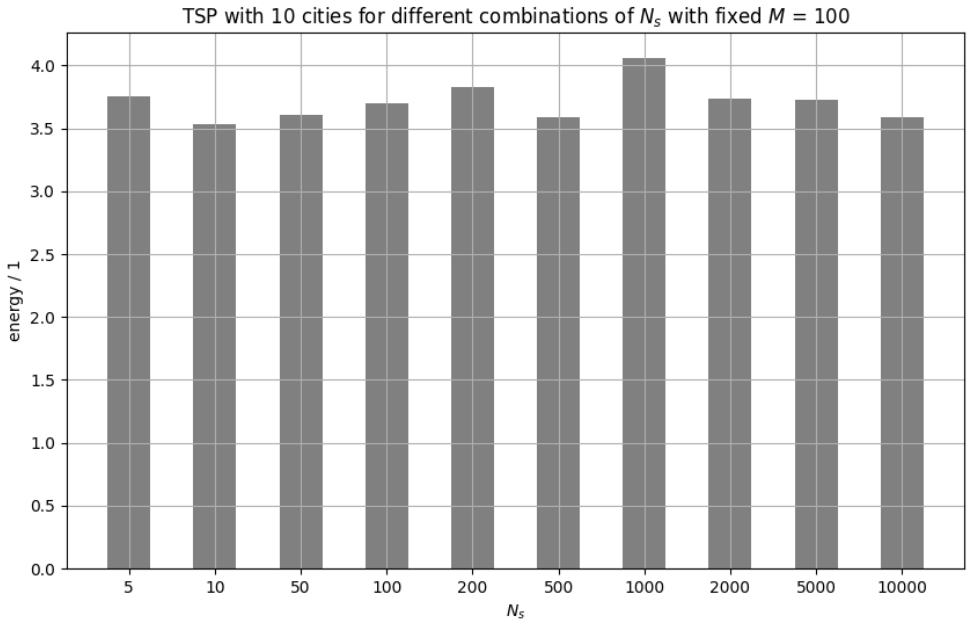

Figure 6 Energy values for different combinations of N_s for fixed $M = 100$ with seed $s = 0$

For the second case where $M = 100$ is fixed and different values for N_s are tested the average energy $\bar{\epsilon}$ and variance $\nu(\cdot)$results to: $\bar{\epsilon} = 3{,}41$ and $\nu(\cdot) = 0{,}041$ [Fig. 6].
We can see that higher M usually give us a lower energy which is not ultimately visible in the case for higher N_s.

Let us visualize this by plotting different combinations of M and N_s for the same number of cities $N = 10$ like it is depicted in figure [7]. Here we can see once again that the optimization algorithm will find different energy minima depending on the iteration length of M and N_s. Here the same seed for sampling the uniform distribution (creating the city map) is once used because of reproducibility. Interesting to note is the overall same path route with same energy $\epsilon = 3{,}5$ for the second row in the plot but with different start / end points.

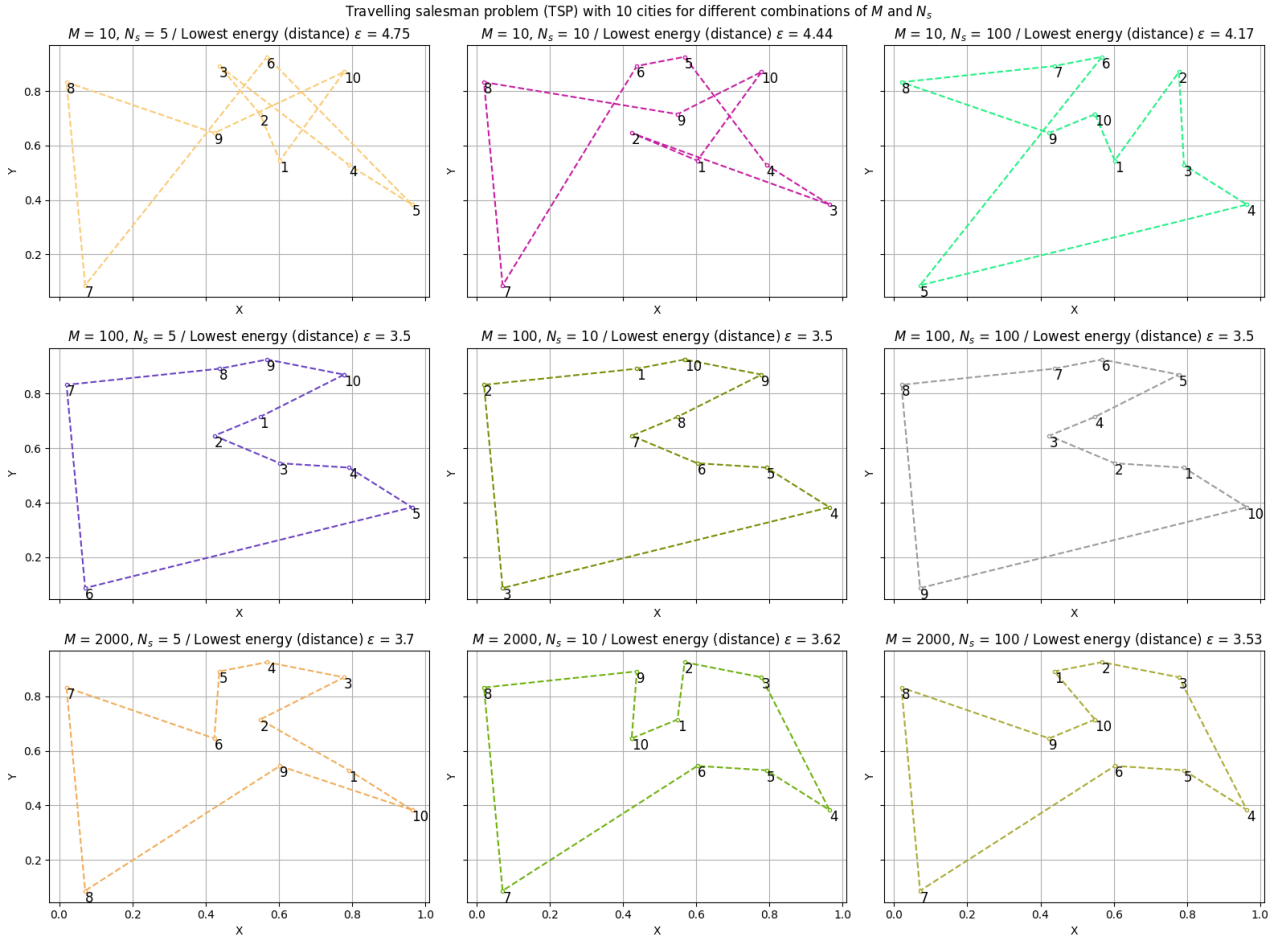

Figure 7 Path plots for different combinations of $M \in [10, 100, 200]$ and $N_s \in [5, 10, 100]$ / random seed $s = 0$ [5]

Additionally we'll look in the following at the average energy $\bar{\epsilon}$ and variance $\nu(\cdot)$ for a fixed M and N_s for different sampled city configurations (without a seed for sampling the city configuration from the uniform distribution) [8 - 9]. The annealing sweep iteration counter M is set to 1000 and the counter N_s is also set to 1000.

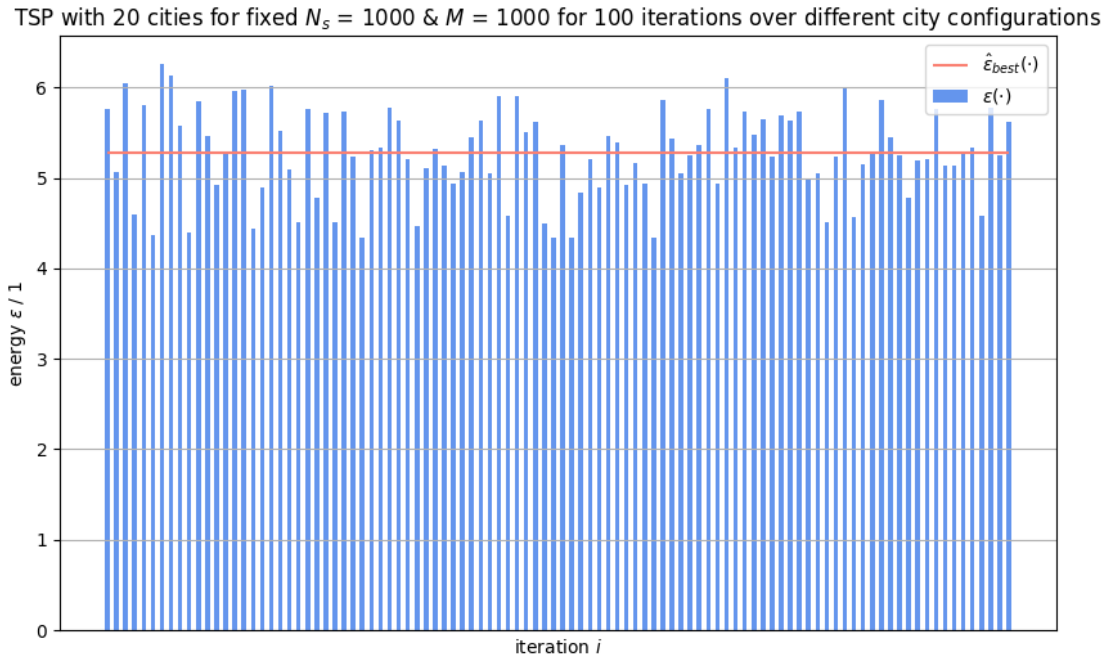

Figure 8 Energy values with average $\bar{\epsilon}_{best}(\cdot)$for $N = 20$ cities for fixed $M = 1000$ and $N_s = 1000$

The average energy provides us here once again an estimator $\bar{\epsilon}_{best}(\cdot)$. In the first case where we take a map with $N = 20$ cities the average energy results to $\bar{\epsilon}_{best}(\cdot) = 5{,}27$ and variance to $\nu(\cdot) = 0{,}24$. In the second case for $N = 40$ the average energy results to $\bar{\epsilon}_{best}(\cdot) = 9{,}23$ and

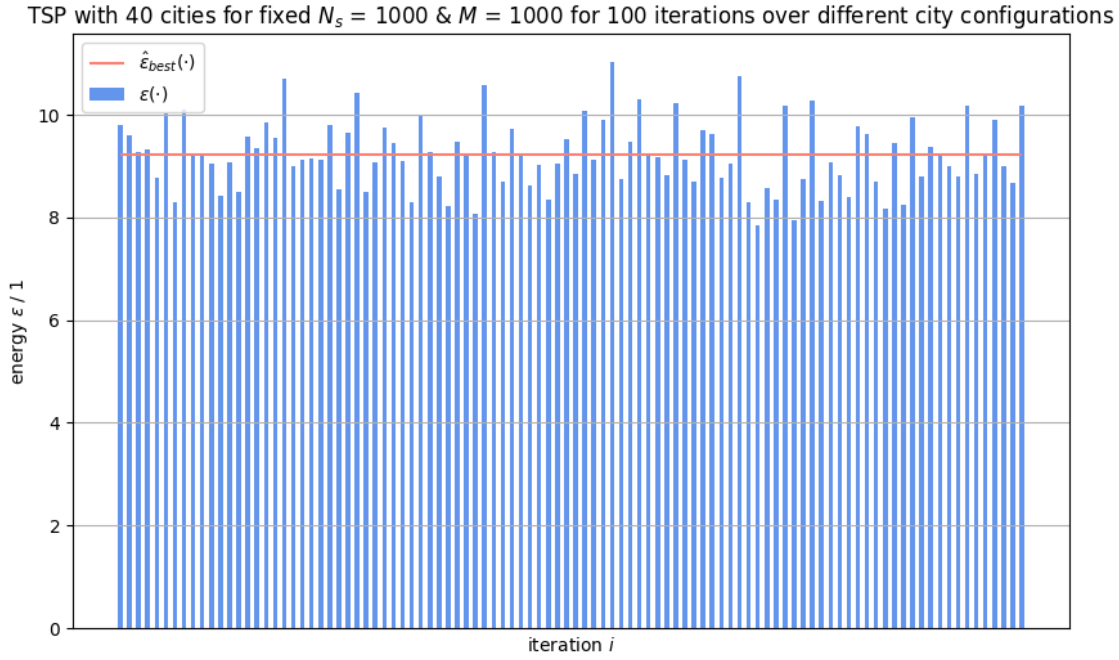

Figure 9 Energy values with average $\hat{\epsilon}_{best}(\cdot)$for $N = 40$ cities for fixed $M = 1000$ and $N_s = 1000$

variance to $\nu(\cdot) = 0{,}45$.
We can see that the overall deviation from the average energy for randomly sampled cities

from a uniform distribution is subsequently low, which means that in average the total distance (energy) which a traveler must go is nearly the same independent on the specific positions of the cities. Or to formulate it in a more mathematical way: Due to the fact that we sample the city locations from a uniform distribution of the (normalized) map the resulting shortest distances (best / lowest energies) are also uniformly distributed over different city combinations.

7 Conclusion

The Traveling Salesman Problem can be relatively simply and elegantly solved with the Metropolis algorithm with Simulated Annealing. For big numbers of city configurations (big N) one has to carefully choose the right parameters for M and N_s for a feasible computation.

Finally, I would like to thank Prof. Dr. Sexty for the very exciting lecture.

8 Code Reference

The corresponding Github repository [2] can be viewed at the following link: MCM Github Repository where the written files for this project are included. The project folder is structured as follows:

```
mcm
├─ sheet_1-6/..
└─ project/
   ├─ main.py
   ├─ tsp_comparison.py
   ├─ tsp_parameter_comparison.py
   ├─ beta_deter.py
   ├─ tsp_multiple_cities.py
   ├─ figures/..
   ├─ tex/..
   └─ requirements.txt
```

References

[1] Diego. "Traveling Salesman Problem (TSP) with Miller-Tucker-Zemlin (MTZ) in CPLEX/OPL". In: *Blog* (2020). URL: https://co-enzyme.fr/blog/traveling-salesman-problem-tsp-in-cplex-opl-with-miller-tucker-zemlin-mtz-formulation/.

[2] Gschaider. "MCM Github Repository". In: (2024). URL: https://github.com/probabilis/mcm.

[3] H.G.Evertz. "Computer Simulations". In: (2020).

[4] University of Hong Kong. "Traveling salesman problem (TSP)". In: (-). URL: https://i.cs.hku.hk/~hkual/Notes/Greedy/Traveling%20salesman%20problem%20(TSP).html.

[5] Numpy. "Random Sampling from Uniform Distribution". In: (2024). URL: https://numpy.org/doc/stable/reference/random/generated/numpy.random.rand.html.

[6] Denes Sexty. "Mone Carlo Methods Projects". In: (2023).

List of Figures