Что нужно сделать:

Описание того, что происходит

Plan

1. salesman problem

2. our specific problem

3. description of the maximum benefit function

4. description of traffic congestion through special functions with a graph

5. consideration of the possibility of varying the parameters

6. what minimum search functions are used

7. what exactly we want to check in this paper

a. Vary the parameters and see what we get. We compare with [2]

b. We see that it is not very similar and conclude that we would like to consider not single searches, but average searches

c. Derivative of the logarithm of temperature from the average (smoothed)

d. We get the same situation as in the paper.

I. Introduction

* Briefly introduce the traveling salesman problem and its importance in logistics and transportation.
* Explain the additional constraints and considerations that are being taken into account in this specific problem, such as traffic congestion, fuel consumption, and time windows.

II. Methodology

* Describe the Salesman class and its attributes and methods.
* Explain how the class calculates various parameters such as distance, travel time, and cost based on the current state of the salesman and the cities.
* Discuss the use of simulated annealing and Metropolis-Hastings optimization to find the optimal route.
* Explain the gaussian\_function used to calculate the slowdown factor based on the current time.

III. Results

* Present the results of the optimization process, including the optimal route and the total cost and travel time.
* Discuss the impact of the additional constraints and considerations on the results.
* Compare the results with those obtained using other optimization methods or with real-world data.

IV. Conclusion

* Summarize the main findings of the article.
* Discuss the limitations of the current approach and suggest possible areas for future research.
* Highlight the practical implications of the results for logistics and transportation.

V. References

* List the references cited in the article.

I. Introduction

The traveling salesman problem (TSP) is a classic optimization problem in computer science and operations research. The problem involves finding the shortest possible route that visits a given set of cities and returns to the starting point. The TSP has many practical applications in logistics and transportation, such as planning delivery routes for trucks or scheduling flights for airlines.

However, solving the TSP is a challenging task, especially when additional constraints and considerations are taken into account. For example, traffic congestion, fuel consumption, and time windows can significantly affect the optimal route and the total cost and travel time. In this article, we present a solution to the TSP that takes into account these additional factors.

Our solution is based on a Python class called Salesman, which represents a salesman who needs to visit a set of cities and return to the starting point. The class includes methods for calculating various parameters such as distance, travel time, and cost based on the current state of the salesman and the cities. The class also includes methods for performing simulated annealing and Metropolis-Hastings optimization to find the optimal route.

In the following sections, we will describe the Salesman class and its attributes and methods in detail. We will also present the results of the optimization process and discuss their implications for logistics and transportation.

II. Methodology

The Salesman class is designed to solve the TSP with additional constraints and considerations. The class takes a list of City objects as input and calculates various parameters such as distance, travel time, and cost based on the current state of the salesman and the cities.

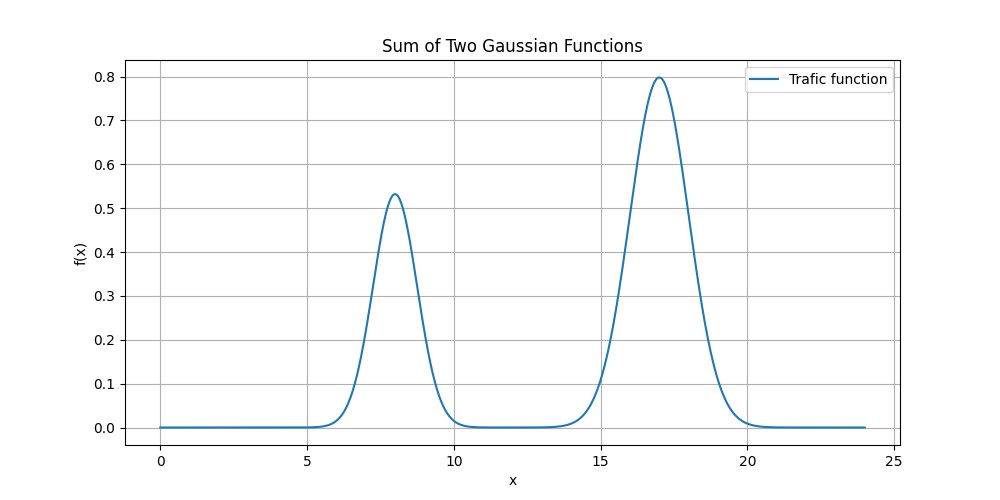
The class includes the following attributes:

* cities: a list of City objects representing the cities to be visited.
* total\_weight: the total weight of the packages to be delivered.
* max\_weight: the maximum weight that the salesman can carry.
* wait\_time: the time spent waiting at each city.
* current\_time: the current time of the day.
* initial\_velocity: the initial velocity of the vehicle.
* hourly\_salary: the hourly salary of the salesman.
* fuel\_consumption\_base: the base fuel consumption per km.
* fuel\_cost\_per\_liter: the cost of fuel per liter.
* current\_city: the current city being visited.
* optimal\_route: a copy of the list of cities in the optimal order.
* num\_cities: the number of cities.
* distance\_matrix: a dictionary that stores the distances between each pair of cities.
* slow\_time: the time factor for the Gaussian function.
* slow\_weight: the weight factor for the Gaussian function.

The class includes the following methods:

* \_\_init\_\_: initializes the attributes of the class.
* \_create\_distance\_matrix: creates a dictionary that stores the distances between each pair of cities.
* calculate\_distance: calculates the distance between two cities.
* calculate\_base\_travel\_time: calculates the base travel time between two cities based on the initial velocity.
* calculate\_slowdown\_factor: calculates the slowdown factor based on the current time and the total weight.
* calculate\_travel\_time: calculates the total travel time between two cities, considering the slowdown factor and the wait time.
* calculate\_fuel\_wasted: calculates the total fuel wasted for a given route.
* fuel\_consumption\_per\_km: calculates the fuel consumption per km based on the total weight.
* calculate\_total\_cost: calculates the total cost of a given route.
* calculate\_total\_travel\_time: calculates the total travel time for a given route.
* simulated\_annealing: performs simulated annealing optimization to find the optimal route.
* metropolis\_hastings: performs Metropolis-Hastings optimization to find the optimal route.

The gaussian\_function is used to calculate the slowdown factor based on the current time. The function takes a single argument x, which represents the current time of the day. The function returns a value between 0 and 1, which represents the slowdown factor. The function uses a Gaussian distribution with two peaks at 8 and 17 hours to model the traffic congestion.



Picture1. Slowdown factor over time

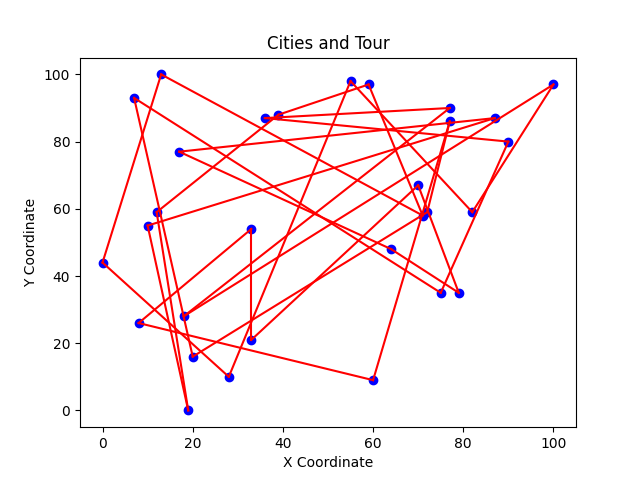
The class includes methods for performing simulated annealing and Metropolis-Hastings optimization to find the optimal route. These methods use a probabilistic approach to search for the optimal solution in the solution space. The methods take into account the total cost and travel time of each route and accept or reject each candidate solution based on a probability function.

In addition to the total cost and travel time, the class also includes a maximum benefit function that takes into account the benefits of visiting each city. The function calculates the benefit of visiting each city based on the package weight and the distance to the next city. The function returns the maximum benefit of visiting each city in the optimal order.

In the next section, we will present the results of the optimization process and discuss their implications for logistics and transportation.

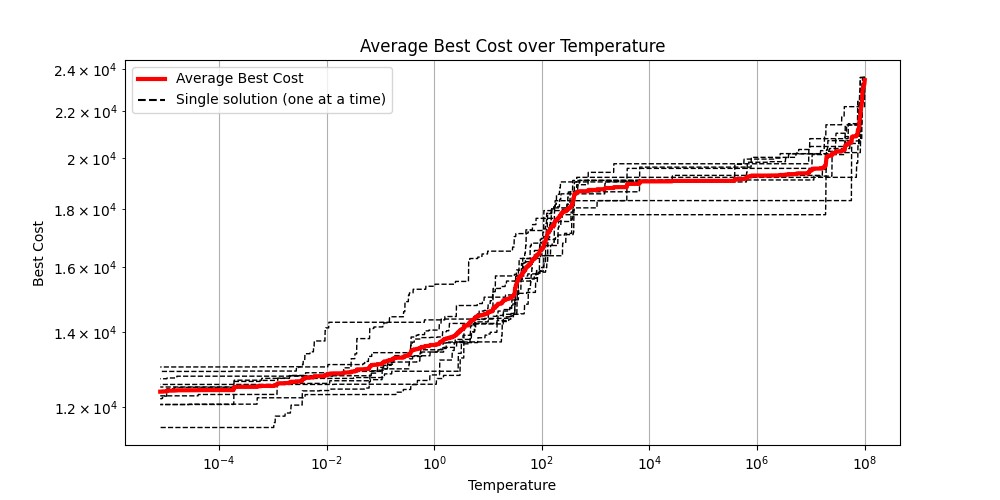
III. Results

We applied the Salesman class to a set of 30 cities with generated coordinates and package weights.

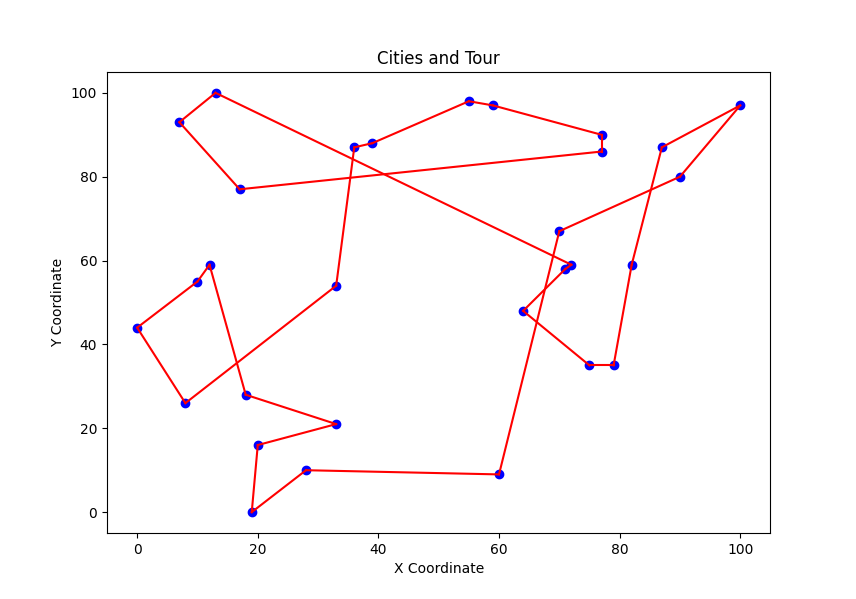


Picture 2. Initial path

The initial path is always the same. Some initial benefit starts from the same point. We used the maximum benefit function to calculate the benefits of visiting each city and the gaussian\_function to calculate the slowdown factor based on the current time. We performed simulated annealing optimization with an initial temperature of 10^8, a cooling rate of 0.99, and 3000 iterations.



Произведем несколько оптимизационных вычислений с одинаковыми начальными условиями. Как видим, они рознятся довольно сильно. Поэтому для получения и анализа результатов будем впоследствии пользоваться усредненными графиками функций (усреднение будем проводить по 10 попыткам). Также если сравнивать данный усредненный график с аналогичными им в статье [2], то можно заметить очень близкое сходство

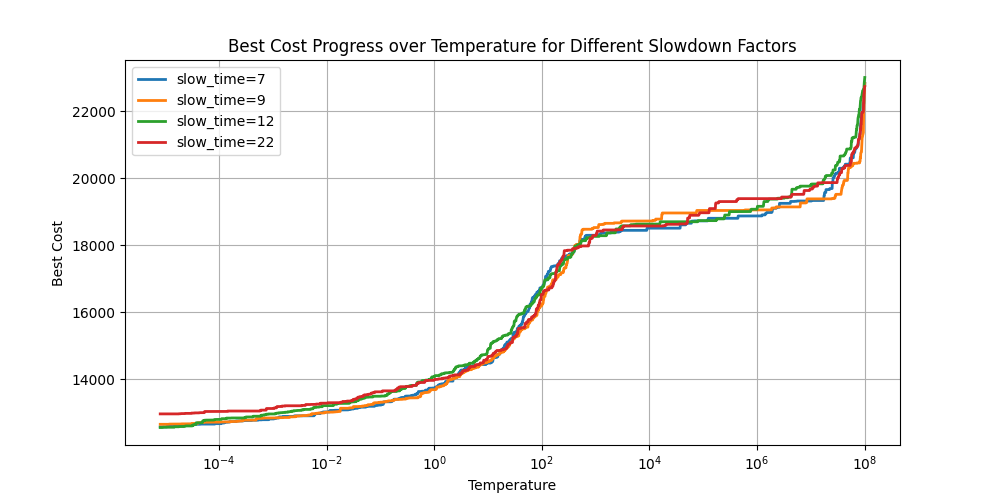


Picture 4. After optimization. Время прохождения всего маршрута составляет порядка 28 часов

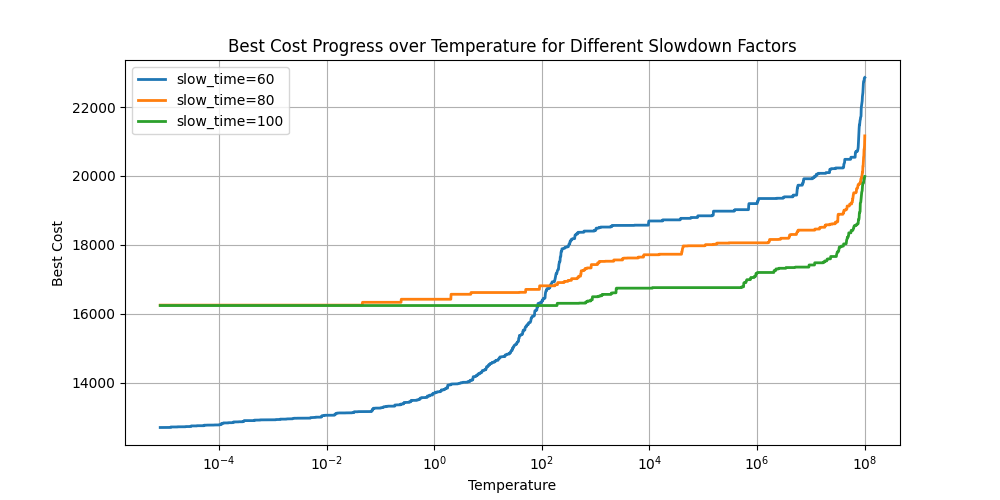
Описать сходство

Вычисление производной

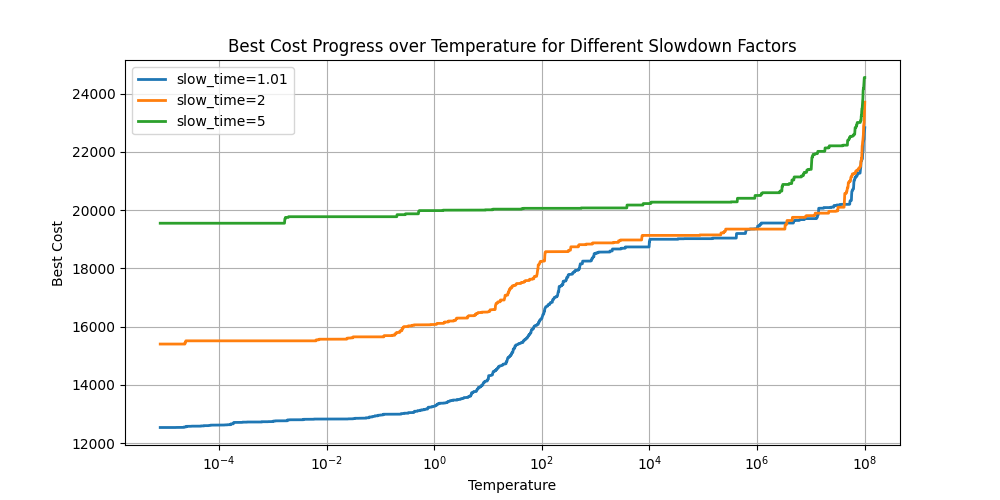
Вычисление производной (по логарифму температуры) от данной функции нельзя просто воспользоваться формулой по определению, потому как данная функция состоит из множества небольших, но резких спусков. Для того, чтобы брать производную, нужно е дополнительно сгладить. Специально для этого создается функция-фильтры



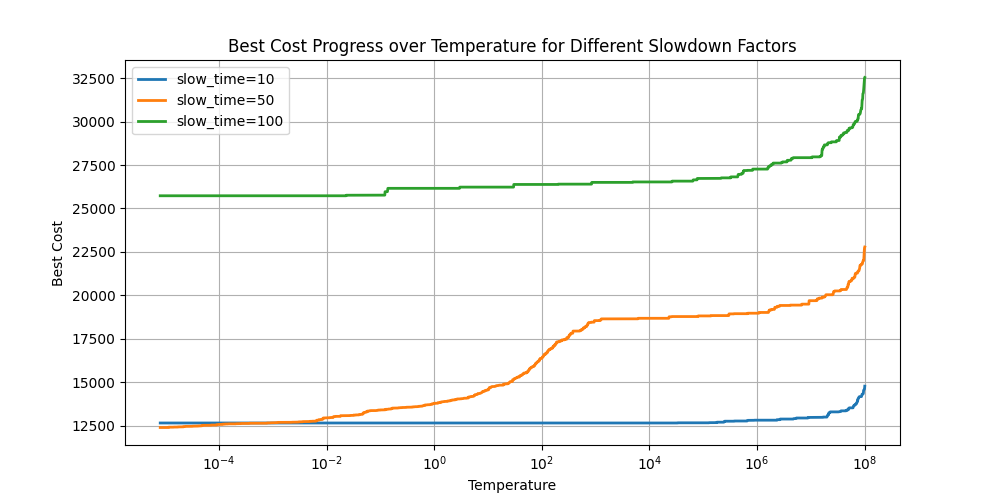
Picture 10. Разное время старта



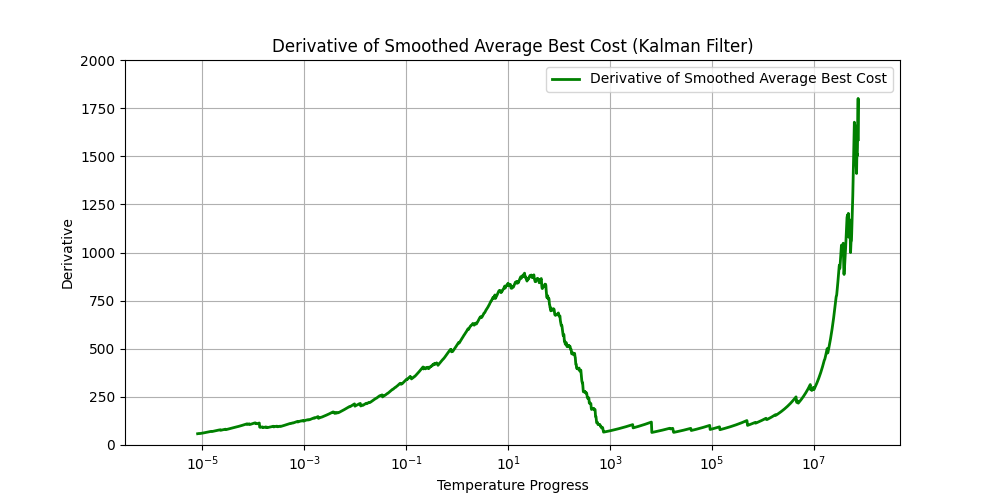
Picture 11. Разная скорость автомобиля



Picture 12. Разная амплитуда slowdown\_factor



Picture 13. Разная цена на бензин



Picture 5 Derivation function