Student ID:82223679

Master's Thesis
Academic Year 2024

# Reducing vehicles in taxi fleet utilizing carpooling with simulated annealing

Graduate School of Science and Technology,
Keio University

Glenn LOUÉDEC

A Master's Thesis submitted to Graduate School of Science and Technology, School of Science for Open and Environmental Systems, Keio University in partial fulfillment of the requirements for the degree of MASTER of Science in Engineering

Thesis Advisor:
Professor Sigeno Hiroshi (Supervisor)

Abstract of Master's Thesis of Academic Year 2024

# Reducing vehicles in taxi fleet utilizing carpooling with simulated annealing

Category: Science / Engineering


## Summary

Nowadays, cities are congested and have a lot of vehicles, 95 percent of the time these vehicles are not even used according to Standing still study. There is different ways to reduce the amount of vehicles in a city but this research plan to focus on carpooling. This method of transportation has existed for a long time and used to be less popular as people prefer the private space of their own car. But recently it started to become popular again with Uber and other applications. The result is that most of the cars driven every days have only one person inside of it. Buses might be a solution but they are not dynamic enough, that is why this research focuses on using autonomous taxis instead. They are much smaller and flexible for this kind of use. This study focuses on reducing the number of vehicles while maintaining time efficiency using an enhanced simulated annealing to find the global optimum faster than other solutions. It will be compared to SUMO greedy shared algorithm. This will be done with different load of passengers, using a simple network as a reference and New York open data as more realistic data. As expected global travel time increased by 30% but with small area and few trips to handle simulated annealing reduces the number of vehicles more than 30% of the fleet size from the simple greedy shared algorithm.



Keywords:

Network, Smart City, Taxi, Carpooling, Optimization



Graduate School of Science and Technology, Keio University

Glenn LOUÉDEC

# Contents

# List of Figures

# Chapter 1

# Introduction

Urban congestion is a growing problem in modern cities, with an excessive number of vehicles contributing to increased pollution, longer travel times, and higher operational costs [1]. In Paris, for instance, traffic congestion accounts for an average of 138 hours of lost time per capita annually (2022) [2]. Traditional approaches to mitigate this issue, such as promoting public transportation or carpooling, have shown limited success due to the preference for the convenience and privacy of personal vehicles. Recent advancements in autonomous vehicle technology, however, offer new opportunities to address these challenges effectively.

This research leverages a modified version of simulated annealing to address multiple objectives in optimizing urban taxi systems. The primary goals are to reduce the number of vehicles required while maintaining high levels of service efficiency. Simulated annealing is particularly well-suited for this optimization challenge due to its ability to explore a vast search space and escape local optima, thus providing near-optimal solutions in complex scenarios. Dynamically optimizing the allocation and routing of autonomous taxis aims to minimize the fleet size required to meet passenger demand without compromising travel times.

This study validates the optimization model by comparing its performance against conventional methods using real-world taxi data from New York city's open data [3]. This comparison allows us to demonstrate the practical benefits and detriment of the approach, including reduced vehicle count and impact on op-

eration efficiency. By doing so, the potential of autonomous taxis to revolutionize urban transportation, making it more sustainable and efficient, is highlighted.

Chapter 2 reviews related work on urban congestion and optimization methods. Chapter 3 details the methodology of the modified simulated annealing approach, including the design of the optimization model. Chapter 4 will present the experimental setup and real-world data used for validation. Chapter 5 discusses the results of the comparison, highlighting the performance improvements achieved by the model. Finally, Chapter 6 concludes the thesis, summarizing results and suggesting directions for future research.

# Chapter 2

# Related Research

Research in the field of smart transportation and autonomous vehicle systems has been extensive, focusing on various optimization aspects such as energy efficiency, resource utilization, and traffic management. However, a critical analysis reveals a gap in the specific focus on reducing the number of vehicles while maintaining effective service levels. This chapter compares notable studies to highlight this gap and set the stage for the proposed approach.

## 2.1. Optimization Frameworks

Zhou et al. (2023) introduce a solar-powered shared electric autonomous vehicle (AV) system, optimizing charging station locations and route planning based on energy distribution and usage patterns [4]. Their model utilizes renewable energy sources to enhance sustainability, addressing key challenges in the deployment of AVs in urban settings. The system's optimization framework considers solar energy availability, energy consumption rates, and travel demand, leading to efficient energy usage. Despite these advancements, this work does not directly address vehicle reduction, focusing instead on the efficiency of energy usage.

Similarly, Pan et al. (2022) propose the Smart Cloud Commuting System (SCCS), which employs shared AVs to enhance vehicle utilization and service efficiency [5]. This system integrates cloud computing with real-time data analytics

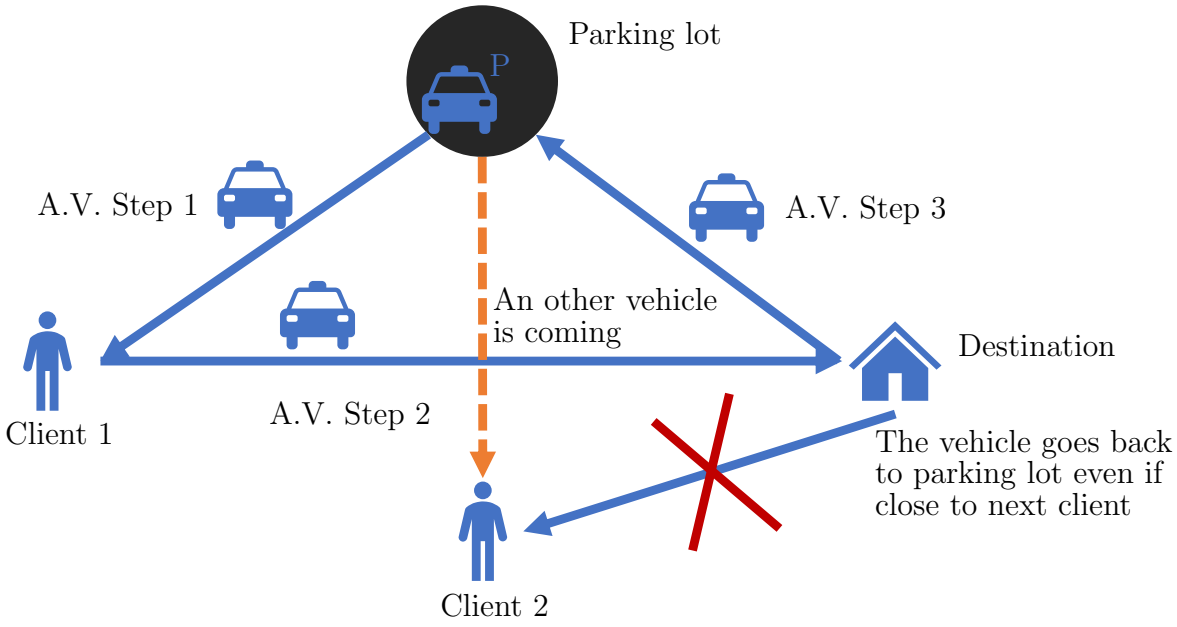


Figure 1: SCSS taking charge of clients individually

to manage and dispatch AVs efficiently as shown on Figure 1. The SCCS demonstrates the ability to serve more trips with fewer vehicles compared to traditional taxi systems. However, the primary focus remains on utilization rather than the outright reduction in vehicle numbers, indicating a partial solution where the emphasis is on optimizing usage patterns rather than the overall fleet size.

## 2.2. Deep Learning and Reinforcement Learning Approaches

Liang et al. (2021) apply deep reinforcement learning to optimize charging schedules for shared electric vehicle (EV) fleets, considering dynamic electricity prices and real-time order information [6]. Their approach leverages advanced machine learning techniques to predict travel demand and optimize charging times, thereby improving the operational efficiency of EVs. This method significantly enhances energy consumption patterns and scheduling efficiency but does not explicitly aim to reduce the fleet size. The focus here is on operational efficiency

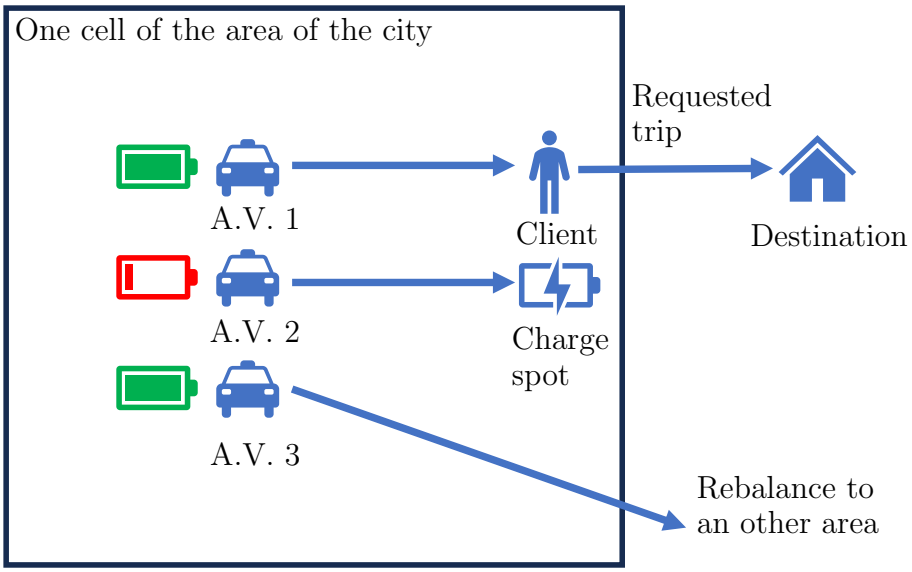


Figure 2: Mobility-aware choices and different states

in terms of energy consumption and scheduling rather than the reduction in the number of vehicles (Figure 2).

## 2.3. Carpooling and Ride-Sharing Optimization

Carpooling and ride-sharing have been explored extensively to enhance transportation efficiency. Zheng and Jiang (2023) develop a multi-objective optimization model for driver-rider matching and route optimization. This model aims to balance platform revenue and path costs [7], improving overall efficiency in ride-sharing platforms (Figure 3). The optimization framework takes into account various factors such as travel time, detours, and passenger preferences, leading to a more efficient matching and routing process. However, the reduction of vehicle numbers is not the central theme. The work primarily addresses the optimization of matching and route planning without specifically targeting fleet size reduction.

Ben Cheikh-Graiet et al. (2020) present DyCOS, a dynamic carpooling optimization system using a Tabu Search-based metaheuristic for real-time ride-matching [8]. DyCOS aims to improve the efficiency of the ride-matching process

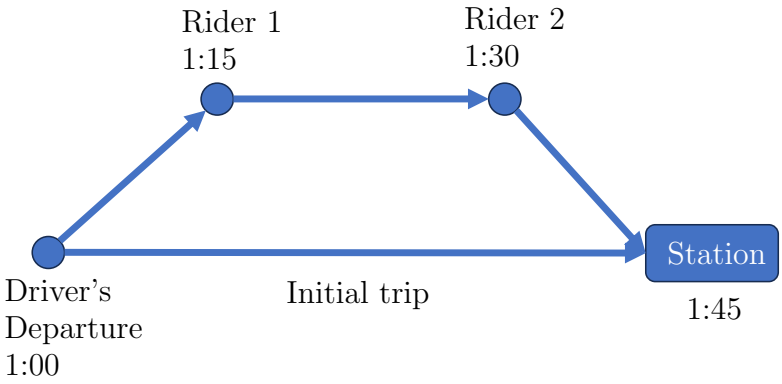


Figure 3: Driver riders example of use

by dynamically adjusting to changes in travel demand and availability of vehicles. The system considers multiple factors such as travel distance, waiting time, and vehicle occupancy rates to optimize the matching process. However, the emphasis remains on the efficiency of matching rather than minimizing the fleet size, indicating a need for solutions focused on vehicle reduction.

## 2.4. Multi-Objective Optimization Algorithms

Su et al. (2022) propose a multiple leaders particle swarm optimization algorithm for the multi-objective fixed crowd carpooling problem. Their approach focuses on minimizing total mileage and extra time consumed, addressing multiple optimization objectives simultaneously [9]. The algorithm considers various constraints such as passenger pickup times, travel distances, and vehicle capacities to optimize carpooling routes. While this work contributes to travel efficiency and time optimization, it does not specifically target vehicle reduction. The primary aim is to enhance carpooling efficiency rather than reducing the number of vehicles required.

## 2.5. Chapter Summary

The comparative analysis of related works reveals a notable gap in the research focus on reducing the number of vehicles in smart transportation systems. While numerous studies have advanced the optimization of various operational aspects, there is limited attention on strategies specifically aimed at minimizing vehicle numbers. The proposed approach leverages simulated annealing to address this gap, aiming to optimize and reduce the fleet size in urban taxi systems without compromising service quality. By focusing explicitly on vehicle reduction, the method differs from existing approaches and fills a critical gap in the literature. The following chapters will delve into the details of the methodology and demonstrate its efficacy through simulation and real-world data analysis.

# Chapter 3

# Conventional Methods

In the domain of optimization for autonomous vehicle systems, several conventional methods have been widely adopted. This chapter discusses the simulated annealing technique and the *SUMO simulation framework* employing the **greedyShared** scheduling algorithm. These methodologies serve as the foundation for the proposed approach and provide a benchmark for comparison.

## 3.1. Simulated Annealing

*Simulated annealing (SA)* is a probabilistic optimization technique inspired by the annealing process in metallurgy. It has been effectively utilized for various optimization problems due to its ability to escape local optima and approach a global optimum [10]. The conventional SA algorithm follows these key steps (Figure 4):

Let $f$ be an objective function that scores a vector $V$ to a value depending on a problem. This function should be minimized for the best solution vector to this problem and maximized for the worst solution vector.

1. **Initialization:** Set an initial vector $S$, an initial temperature $T$, and a cooling rate $C$. This initial vector is by default set as the best vector $V_b$.

2. **Iteration:** Generate a new vector $S'$ by making a small change to the

current vector. This new vector is sometimes called a neighbor of the current vector.

3. **Acceptance Criteria:** Accept the new vector $V'$ if it improves the objective function $f$, i.e., if $f(V') < f(V)$. If not, accept it with a certain probability that decreases with the temperature, e.g., accept if for $r \in [0,1], r < e^{-\frac{f(V')-f(V)}{T}}$.

4. **Cooling Schedule:** Gradually reduce the temperature according to a predefined schedule, $T' = T \times C$ with $T'$ being the next iteration's temperature.

5. **Best Vector:** At each step, if $f(V_b) > f(V')$, then set the new best vector to be $V'$.

6. **Termination:** Repeat the iteration process until the system cools to a stop.

While SA is robust in exploring the solution space and avoiding local optima, it typically focuses on optimizing a single objective function. In contrast, the research aims to extend SA to handle multiple objectives, specifically targeting the reduction of vehicle numbers in a fleet while maintaining service quality.

## 3.2. SUMO Simulation and GreedyShared Scheduling Algorithm

*Simulation of Urban MObility (SUMO)* is an open-source traffic simulation suite that allows for modeling and simulating urban transportation systems [11]. It provides a comprehensive environment for evaluating traffic management strategies and vehicle dispatch algorithms.

One of the conventional methods employed in SUMO for taxi dispatching is the **greedyShared** scheduling algorithm [12]. The **greedyShared** algorithm, implemented within the **drtOnline.py** tool, is utilized for ride-pooling and ride-sharing services, optimizing the allocation of vehicles to passengers in real-time

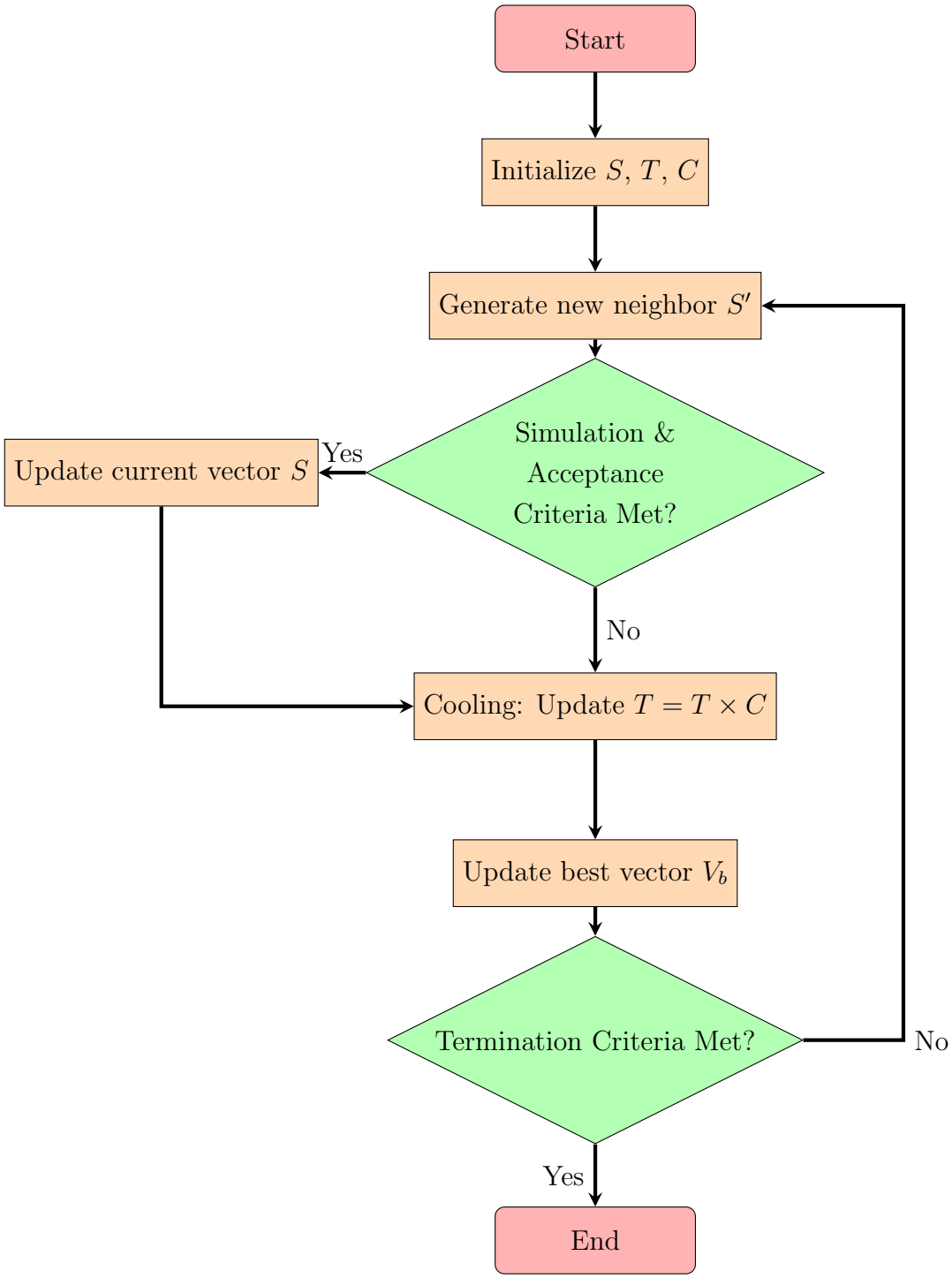


Figure 4: Flowchart of the Simulated Annealing Algorithm

scenarios. This method serves as a benchmark for evaluating the efficiency of the proposed simulation approach [13].

This algorithm operates as follows (Figure 5):

1. **Request Matching:** Incoming ride requests are matched to available taxis based on proximity and current load.

2. **Greedy Dispatching:** Taxis are dispatched in a greedy manner, selecting the closest available taxi to serve a new request.

3. **Route Optimization:** The selected taxi's route is updated to include the new passenger's destination, optimizing for the shortest path using *integer linear programming (ILP)* [14].

The **greedyShared** scheduling algorithm is straightforward and efficient, often yielding good performance in terms of response time and resource utilization. It employs a different optimization approach using ILP. However, it does not explicitly aim to minimize the number of vehicles in operation, which is a critical objective in the study.

## 3.3. Critique of Conventional Methods

While both simulated annealing and the greedyShared scheduling algorithm provide effective solutions within their respective contexts, they fall short in addressing the specific challenge of reducing fleet size. Here is a detailed critique of these methods:

### 3.3.1 Simulated Annealing

Single Objective Focus: Traditional SA is designed to optimize a single objective function, which limits its application in scenarios where multiple objectives need to be balanced [15].

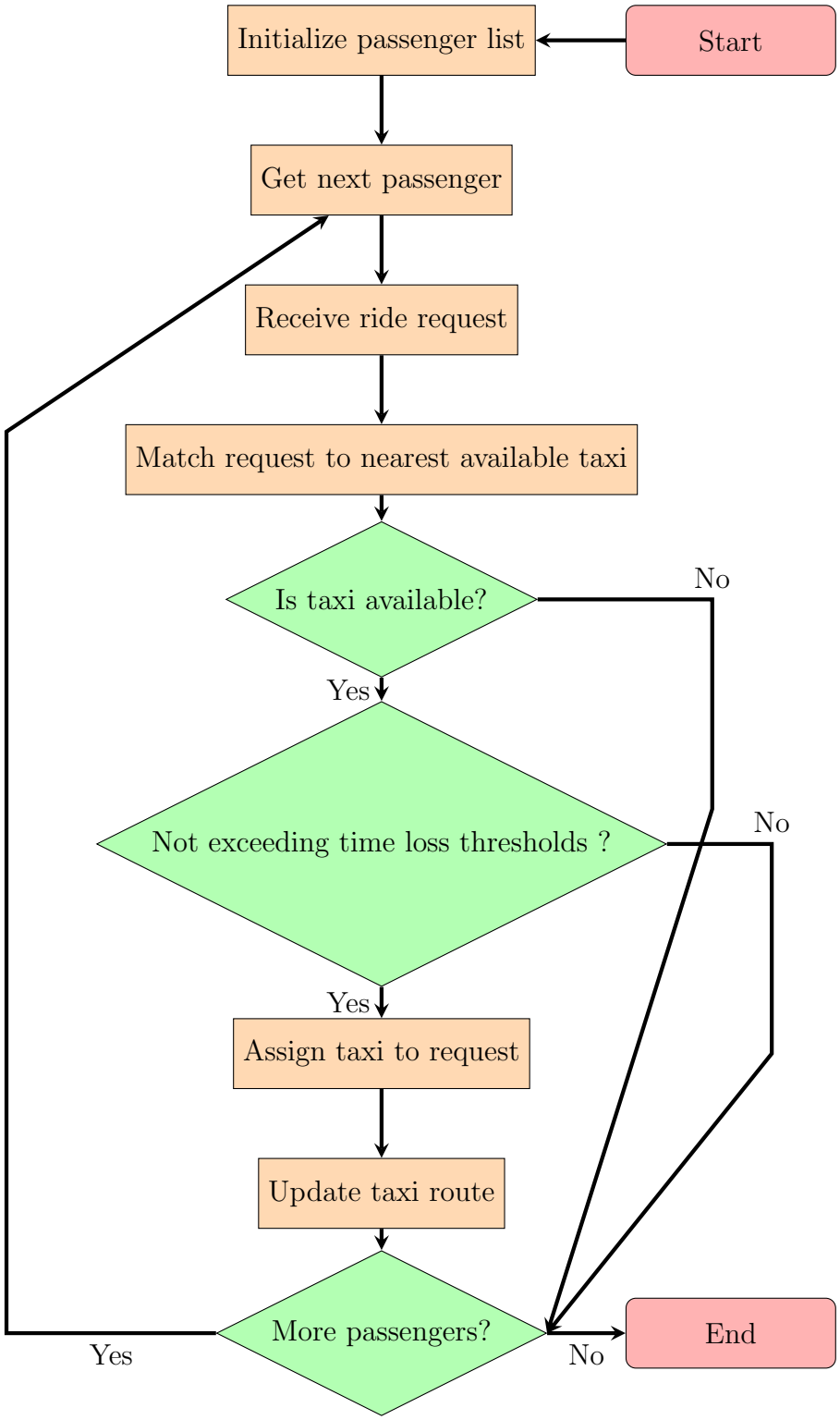


Figure 5: Flowchart of the GreedyShared Scheduling Algorithm in drtOnline.py

Efficiency Concerns: Although SA can escape local optima, it may require a significant amount of computational time to converge to a global optimum, especially for large-scale problems.

Adaptation Complexity: Extending SA to handle multiple objectives and integrate real-time data for dynamic dispatching can be complex and computationally intensive.

### 3.3.2 GreedyShared Scheduling Algorithm

Short-term Optimization: The greedy approach focuses on immediate efficiency, selecting the closest available taxi without considering long-term fleet optimization.

Lack of Global Perspective: This method optimizes routes on a per-request basis, which can lead to suboptimal fleet management in the long run.

Vehicle Utilization: While it improves response times and resource utilization, it does not aim to reduce the number of vehicles in operation, which is crucial for urban congestion and environmental impact.

## 3.4. Chapter Summary

While both simulated annealing and the greedyShared scheduling algorithm provide effective solutions within their respective contexts, they fall short in addressing the specific challenge of reducing fleet size. Simulated annealing's traditional single-objective focus and the greedyShared algorithm's emphasis on immediate efficiency highlight the need for a more holistic approach. The proposed method aims to bridge this gap by integrating multi-objective optimization into the dispatching framework, prioritizing both service efficiency and vehicle reduction. The following chapters will present the methodology, leveraging the strengths of these conventional methods while addressing their limitations to achieve the research objectives.

# Chapter 4

# Proposed Method

This chapter will propose a method for multi-objectives simulated annealing with threshold. First, the problem setting will be reviewed then the process to get new neighbors for the simulated annealing will be checked as the vector used are quite special and then to finish the multi-objective simulated annealing with threshold will be presented.

## 4.1. Problem Setting

### 4.1.1 Objective

The main objective of the proposed method is to demonstrate that it is possible to achieve good performance, specifically aiming for a 10% increase in total waiting and travel time, while also reducing the number of vehicles required.

### 4.1.2 Variables and Parameters

- **Temperature** $T$: The control parameter for simulated annealing, which decreases over time.
- **Cooling rate** $C$: The rate at which the temperature decreases.

- **Iteration limit** $It$: Maximum number of iterations for the simulated annealing.
- **Taxi id** $v$: Identifier for each taxi.
- **Number of taxis** $v_{nb}$: Total number of taxis available.
- **Person id** $p$: Identifier for each person.
- **Number of persons** $p_{nb}$: Total number of persons requesting rides.
- **Vector** $V$: A dictionary that assigns a list of pickup/drop-off points of persons to a taxi, such as $V = \{v_1 : [p_1, p_1, p_2, p_2, ...], v_2 : [...], ...\}$. The first occurrence of a person id represents the pickup, and the second occurrence represents the drop-off.
- **Initial Vector** $V_i$: Vector that had one passenger per vehicle $V_i = \{v_1 : [p_1, p_1], v_2 : [p_2, p_2], ..., v_{v_{nb}} : [p_{v_{nb}}, p_{v_{nb}}]\}$.
- **Occupation Score** $f_{os}(V)$:

$$f_{os}(V) = (v_{nb} - \sum_{v} \delta_{V[v]=[\,]}) \times \frac{p_{nb} + w_p + \frac{d_p}{2}}{p_{nb}},$$

  where $w_p$ and $d_p$ are the number of persons still waiting and driving at the end of the simulation, respectively, and

$$\delta_{V[v]=[\,]} = \begin{cases} 1, & \text{if } V[v] \text{ is empty}[\,] \\ 0, & \text{else} \end{cases}.$$

  To summarize, the occupation score is the number of used vehicles.

- **Time Score** $f_{ts}(V)$:

$$f_{ts}(V) = \sum_{p} T_p,$$

  where $T_p = W_p + D_p$, $W_p$ is the waiting time of person $p$, and $D_p$ is the driving time of person $p$.

### 4.1.3 Input Data

The evaluation uses open data from New York City's green taxi fleet from 2015 [3], which provides information on start and end positions, times, and the number of passengers. Here is the full entry:

- vendorid: Text
- pickup_datetime: Floating Timestamp
- dropoff_datetime: Floating Timestamp
- Store_and_fwd_flag: Text
- rate_code: Number
- Pickup_longitude: Number
- Pickup_latitude: Number
- Dropoff_longitude: Number
- Dropoff_latitude: Number
- Passenger_count: Number
- Trip_distance: Number
- Fare_amount: Number
- Extra: Number
- MTA_tax: Number
- Tip_amount: Number
- Tolls_amount: Number
- Ehail_fee: Number

- Improvment_surcharge: Number
- Total_amount: Number
- Payment_type: Number
- Trip_type: Number

### 4.1.4 Optimization Goals

The primary optimization goal is to minimize the number of vehicles by maximizing the number of passengers per vehicle. A low score (indicating better performance) is achieved when taxis have high occupancy or remain unused, while a high score (indicating worse performance) occurs when taxis have low occupancy and are mostly empty. The secondary optimization goal is to minimize the combined total waiting and travel times for all passengers.

### 4.1.5 Comparison Benchmark

The proposed method is compared against the greedyShared algorithm included in SUMO, evaluating performance based on the same scoring function. The implementation involves using the SUMO simulation environment via the TraCI library interface. The process begins with data retrieval from the open data API, followed by data processing and simulation setup. The fitness function evaluates each candidate solution, and after a set number of iterations, the best solution is determined.

### 4.1.6 Limits

The problem setting is simplified to accommodate the constraints of the available data and the simulation environment.

- **Input Data:** All taxi rides are initialized at the beginning of the simulation, regardless of their actual start times, and vehicles start at passenger pick-up positions due to the provided data structure.

- **Traffic congestion:** The simulations do not account for traffic congestion or real-time traffic conditions.
- **Taxi paths:** The vector only dictates the order of pickup and drop-off; the routing of taxis between pickup/drop-off points is computed by SUMO's built-in fastest route algorithm.
- **Data sampling:** To speed up the simulation time, a sample rate lower than the speed of the simulation is used. This results in slightly inaccurate values for the scoring of the vector, but due to consistency, this limit has a low impact.

### 4.1.7 Challenges and Considerations

One of the major challenges is ensuring the optimization runs efficiently. To address this, certain aspects of the problem are simplified and a multi-objective approach within the simulated annealing framework is adopted. Another challenge is managing the dual optimization goals, which is handled by appropriately weighting and balancing the objectives within the optimization process. The neighbors generation is also changed as the vector is not continuous and it has to dimension of changes possible: order of passenger in a vehicle and assignment of passenger to vehicle.

## 4.2. Neighbors in Simulated Annealing

This section propose modifications to the Simulated Annealing algorithm specifically for the creation of neighbors. The key ideas behind these modifications are aimed at finding a way to get discrete neighbors, enhancing the exploration and exploitation phases of the algorithm, thereby improving its overall performance.

### 4.2.1 Neighbor Creation

Now that explanation of the setting of this problem has been made, let's discuss the first proposed change in Simulated Annealing, which involves the creation of neighbors.

1. **Adaptive Number of Changes:** First, set a number of changes in the current vector[1] according to the temperature using the following formula:

   $$nb_{changes} = v_{nb} \times e^{\frac{T}{T_i}-1}, \text{ where } T_i \text{ is the initial temperature.}$$

   This means that the maximum number of changes is $v_{nb}$, so one for each vehicle. This adaptive approach helps to rapidly explore a diverse set of solutions at higher temperatures and progressively focus on fine-tuning as the temperature decreases.

2. **Randomized Modification Strategy:** Then, for each change, randomly select whether to swap persons within a vehicle or change the assignment of a person from one vehicle to another, with equal probabilities of 50% for each. This strategy ensures that the neighbor solutions remain valid by preventing unrealistic scenarios, such as a passenger's pickup and drop-off being assigned to different vehicles.

[1] **Reminder:** $V = \{v_1 : [p_1, p_1, p_2, p_2, ...], v_2 : [...], ...\}$

**Algorithm 1** Get Neighbor Solution

**function** GetNeighbor(current_solution, temperature, initial_temperature)
    *neighbor_solution* ← copy.deepcopy(current_solution)
    $nb\_changes \leftarrow \left\lfloor \text{len(current\_solution)} \cdot \frac{\exp(\frac{\text{temperature}}{\text{initial\_temperature}})}{\exp(1)} \right\rfloor$
    **for** $i \leftarrow 1$ to *nb_changes* **do**
        ModifyPersonAssignment(neighbor_solution)
    **end for**
    **return** *neighbor_solution*
**end function**
**function** ModifyPersonAssignment(neighbor_solution)
    *neighbor_type_rate* ← random.random()
    **if** *neighbor_type_rate* $< 0.5$ **then**
        ChangePersonAssignment(neighbor_solution)
    **else**
        SwapPersonsWithinVehicle(neighbor_solution)
    **end if**
**end function**

### 4.2.2 Key Points of the Proposed Modifications

The first key point is the adaptive number of changes, which facilitates extensive exploration of the solution space at the beginning and concentrates on optimization as the algorithm progresses. This dynamic adjustment enhances the algorithm's ability to escape local optima and find better solutions.

The second key point is the randomized modification strategy, which maintains the validity of solutions and ensures a balanced exploration of different types of neighbor solutions. By preventing infeasible configurations, this approach increases the efficiency and reliability of the Simulated Annealing process.

These proposed modifications aim to leverage the strengths of the Simulated Annealing algorithm while addressing its limitations, thereby improving its applicability and performance in solving the simulation.

## 4.3. Multi-objective Simulated Annealing with threshold

Simulated Annealing doesn't include multi-objectives in its conventional form. It also doesn't include threshold. To add these parts, some changes to the algorithm are made.

- **Fitness function:** The fitness function now returns a list of objectives $f(V) = [f_{os}(V), f_{ts}(V)]$.
- **Threshold:** introducing time score threshold $th = th_{ts} \times f_{ts}(V_i)$ that will be use to permit acceptance of vector with a bit worse time score depending on the threshold.
- **Acceptance Criteria:** Now acceptance of the new vector $V'$ is using all scores. Acceptance if, for

$$r \in [0, 1], r < \text{acceptanceProbability}(V, V', T, th).$$

**Algorithm 2** Acceptance Probability Function with multi-objective and threshold

```
1: function ACCEPTANCEPROBABILITY(current_objectives, new_objectives, temperature, threshold)
2:     if new_objectives[0] < current_objectives[0] AND new_objectives[1] < threshold then
3:         return 1.0
4:     else
5:         return exp(-Σ_i (new_objectives[i] - current_objectives[i]) / temperature)
6:     end if
7: end function
```

- **Best vector:** $V'$ is now new best vector if $f(V')_{oc} < f(V_b)_{oc}$ and $f(V')_{ts} < th$ where $V_b$ is the current best vector.

This new method permit to have multiple objectives and can also control how much permissive the optimization is. As an example with the threshold parameter set to $th_{ts} = 1.1$, it permit the simulation to have 10% increase of the time score of the initial vector (vector were every person has it's own vehicle).

## 4.4. Summary of Chapter

This chapter outlined the proposed method for optimizing taxi assignments using simulated annealing with a multi-objective and threshold approach. Key objectives, variables, and parameters were defined, the input data and optimization goals were explained, and the approach to neighbor creation and acceptance criteria was highlighted. By balancing occupation and time scores, the aim is to enhance taxi usage efficiency and reduce passenger waiting and travel times.

The next chapter will evaluate the performance of the proposed method through detailed simulations and comparisons with existing approaches, demonstrating its effectiveness and practical benefits.

# Chapter 5

# Evaluation

## 5.1. Dataset

Experimentation use two types of road networks. First a simple Manhattan simulated network with random trips as control data and then a network of New York city with data from the 2015 Green Taxi Trip dataset from the Taxi and Limousine Commission of New York City [3]. This dataset includes information about Street-Hail Liveries, also known as green cabs [16], which are for-hire vehicles permitted to accept street-hails. These vehicles operate outside the Hail Exclusionary Zone, which is defined as south of West 110th Street and East 96th Street. Green taxis also accept dispatches from FHV Bases and licensed E-Hail companies.

### 5.1.1 Time Period and Duration

The simulation run with 200 and 500 trips on the simple Manhattan network. Then the New York City open data is typically set to a 10 and 30-minute window, which usually provided 150 and 496 trips respectively. Only the vehicle departing during this window are taken.

The two time windows are taken from 2015 1 January 12:00:00 to 12:10:00 and 12:30:00 respectively.

For the simulation three scenario are used:

- **Scenario 1:** The simple Manhattan network with only 200 trips is used and then the thresholds values of the time score from 1.1 to 1.5 is changes to see the impact on performances with the goal to get better occupation score.
- **Scenario 2:** The simple Manhattan network with random trips set at 200 and 500 with a fixed thresholds value for the time score set at 1.2 is used and then the number of iterations on the simulation is changed to see the effects on the results and find a more optimal number of iterations.
- **Scenario 3:** The information's learned from these first two simulation are used to run the simulation with the New York City network and it's 2015 Green Taxi Trip dataset with 10 minutes and 30 minutes of traffic. Results are compared to the ones using greedyShared algorithm.

### 5.1.2 New York City Dataset Contents

The dataset contains lots of information, here are the information used in the simulation for each trip:

- Start and stop positions (longitude and latitude) of each trip
- Number of passengers
- Start and finish times (pickup and dropoff times)

### 5.1.3 Dataset Access and Format

The New York City dataset is accessed via the Socrata library [17], which allows us to interact with the open data API [18].

### 5.1.4 Preprocessing Steps of the Dataset

The preprocessing steps include:

- Retrieving the data using the Socrata library.
- Filtering the data based on time and geographical position to fit the simulation requirements.
- Converting trip positions into edges for simulation using the TraCI library [19].

### 5.1.5 Sumo Network

SUMO tool OsmWebWizard.py [20] provided the network of the city of New York (Figure 6). Because of calculation limitation the small road are removed and only the main roads of the city are kept.

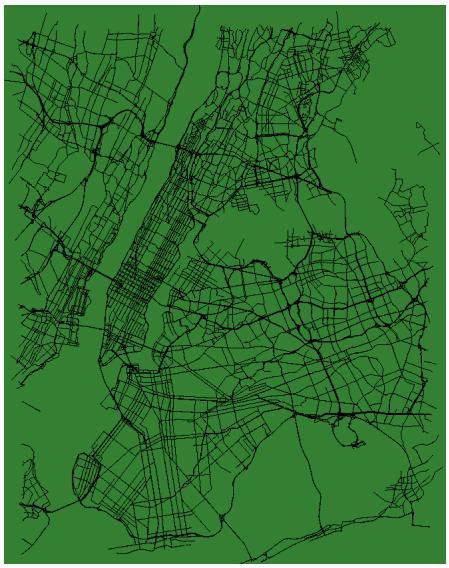

Figure 6: Sumo New York City network

The other network, representing a simplified Manhattan road network, was build using *netgenerate* SUMO command tool [21](Figure 7).

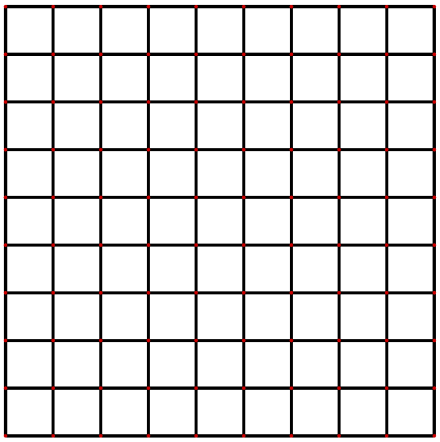

Figure 7: Simple Manhattan network

The First network has dimensions of around 35km and 45km while the second one are 900m and 900m only. The two network difference in size that is really important as it takes more time to travel, the number of steps inside SUMO simulation needs to be higher for the first network. Because of that the execution time of the optimization is drastically increased.

### 5.1.6 Limitations and Challenges

One of the main limitations of using this dataset is the increased computational load associated with larger data volumes. The more data used, the more computational resources are required, which can impact the performance and efficiency of the simulation.

The dataset provides a comprehensive view of green taxi operations in New York City, allowing us to accurately model and simulate various dispatching strategies. This forms the basis for evaluating the proposed method against conventional approaches.

## 5.2. Results

To evaluate the performance of the proposed optimization method, a series of simulations using the simple Manhattan network and New York City network with it's taxi data is conducted. The evaluation focused on two primary metrics: the occupation score [1] and the time score [2]. For comparison, the greedyShared scheduling algorithm is used as a baseline.

For all this part the initial temperature is set to $T_i = 1000$, and the cooling rate is set to $C = 0.95$. The maximum number of steps for the simple Manhattan network simulation in SUMO is set at 700 and for the New York City network it is set at 3000. The number of trips will represent the number of person and the number of vehicles that can be used.

### 5.2.1 Effect of threshold on score

This subsection will look at the effect of the time thresholds $th_{ts}$ on both results score. Simulation will be run with 20 iterations of the simulated annealing and 200 trips.

The results in Figure 8 show that the best occupation score and reduced the number of vehicle is reached for $th_{ts}$ around 1.3, however the time score is impacted. With high threshold value both score get worse as the acceptance is more permissive and this leads to less convergence. For the rest of the experiments the value of 1.3 will be used as time threshold.

[1] depends on the number of vehicles used
[2] total travel + waiting time

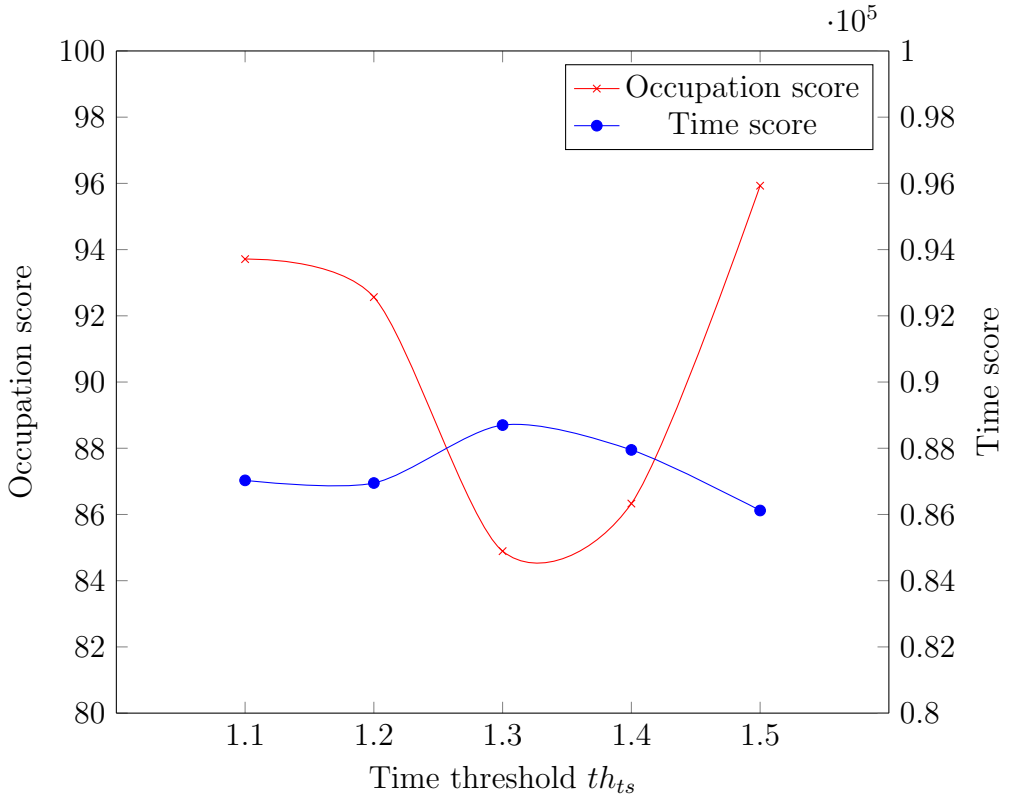


Figure 8: Effect of thresholds on the scores using multi-objective simulated annealing on the simple Manhattan network and 200 random trips and 20 iterations

### 5.2.2 Optimal number of iterations

This experiment will use the simple Manhattan network again, setting the thresholds according to previous part.

100 iterations and 20 iterations with 200 and 500 trips are used and compared to the results with greedyShared algorithm to see if high number is necessary to get good results.

In Figure 9 and Figure 10 with 200 or 500 trips the same kind of results are shown. The difference in results between 100 and 20 iterations is also very small. 20 iterations will be used for the following simulations as it is way faster.

In general greedyShared if way faster to execute than the proposed method, in fact the algorithm in included in SUMO whereas the proposed algorithm uses

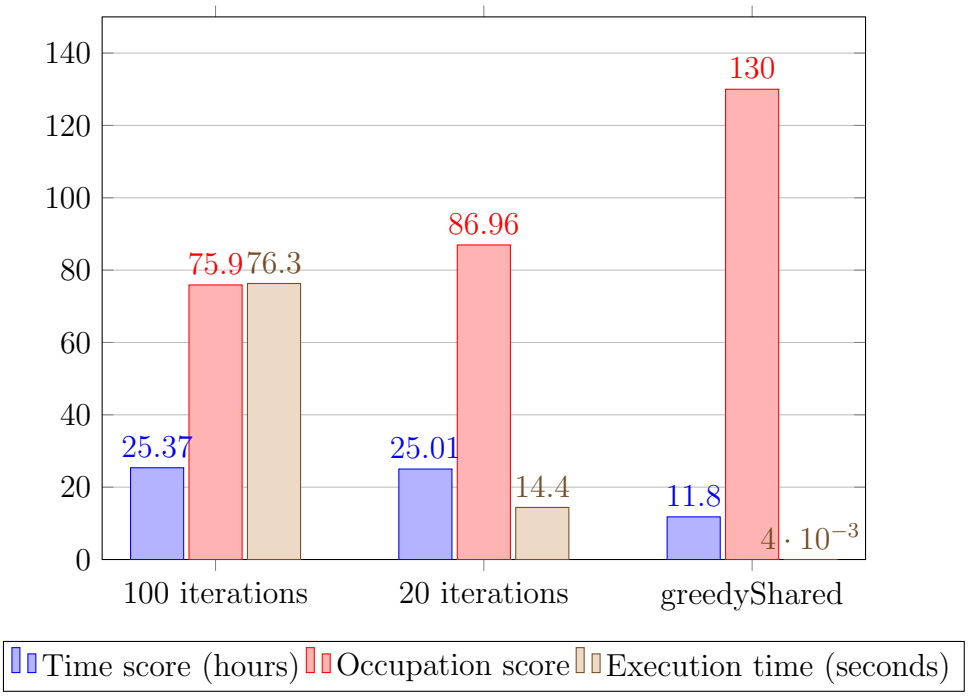


Figure 9: Score comparison depending on the number of iterations with 200 trips using the simple Manhattan network

SUMO API which increase the computation time. But greedyShared algorithm doesn't reduce as much the number of vehicle, in fact it prioritize the time efficiency and halves the time score in both cases.

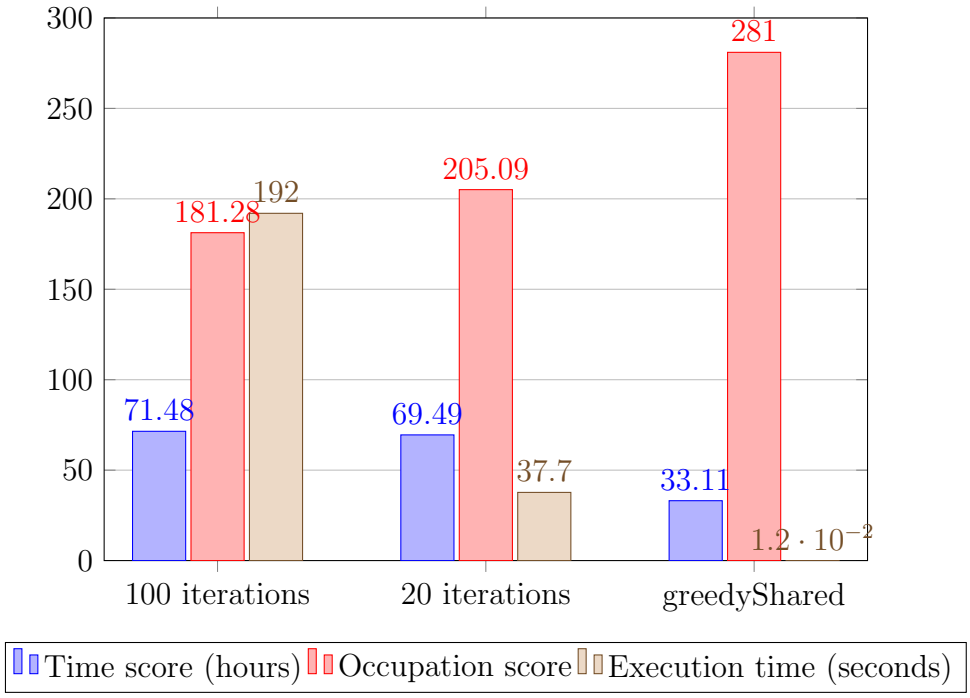


Figure 10: Score comparison depending on the number of iterations with 500 trips using the simple Manhattan network

### 5.2.3 Results with real data

This simulation tested, with the previous information, the effectiveness of the proposal with the New York City network and the real taxi data. The results are compared with the greedyShared algorithm and the New York dataset. From the previous two sections it has been decided to use a time threshold $th_{ts} = 1.3$ and a number of iterations for the simulated annealing of 20 iterations. First the simulation will run with with 10 minutes worth of taxi trip ordeal (150 trips) and then 30 minutes (496 trips).

In the Figure 11 and Figure 12, the proposed method is far less efficient on this network than the greedyShared in term of reducing the number of vehicles as it doesn't even improve the occupation score compared to the real data. In fact the size of the road network might be the issue here, it also causes increased execution time for greedyShared that is surprising.

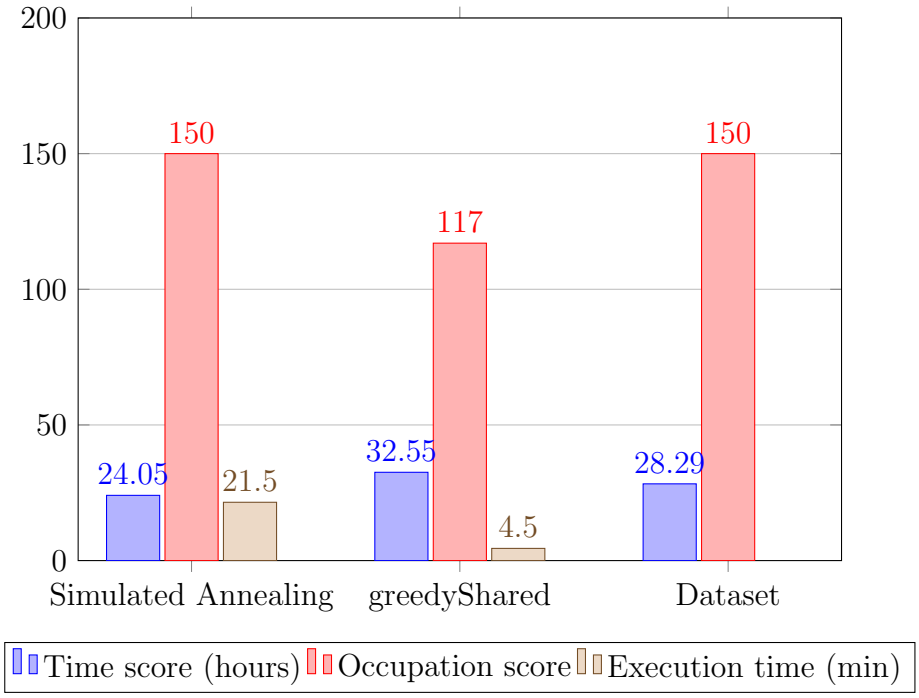


Figure 11: Comparison of proposed method, greedyShared and real dataset using 10 minutes worth of data from the dataset and the New York City network

On the other hand, when comparing the time score of the two methods with the real data, the results are quite close which is unexpected. The issue here is the lack of any traffic except the taxi, this assumption really improved the results compared to the real time data and thus is not realistic.

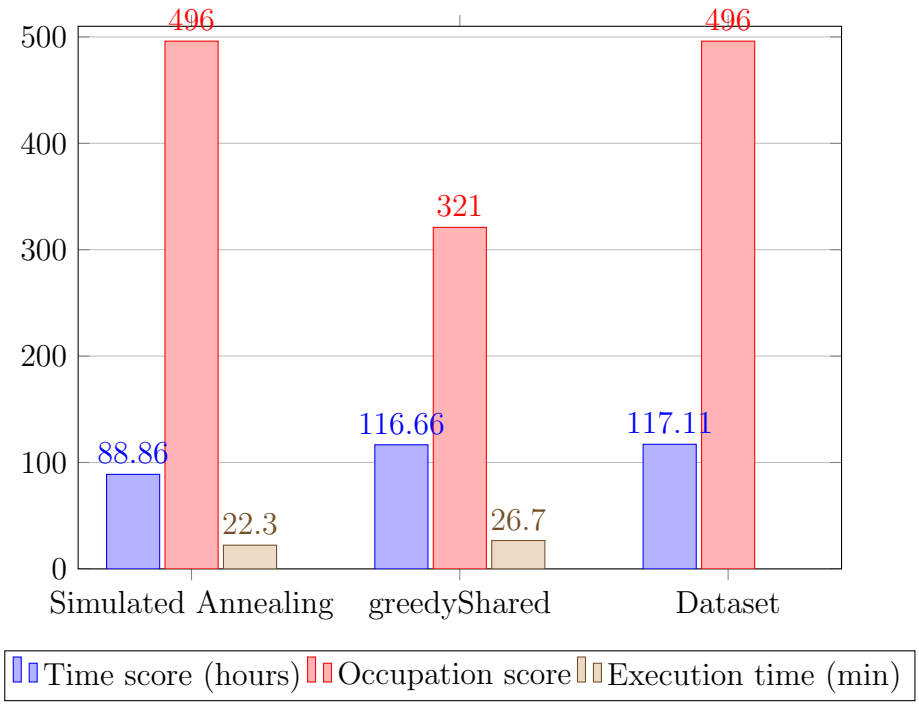


Figure 12: Comparison of proposed method, greedyShared and real dataset using 30 minutes worth of data from the dataset and the New York City network

## 5.3. Summary of Chapter

The results indicate that the proposed method is not effective using a big network at the scale of a city but greedyShared is only a bit better. However on a smaller scale network it can achieve better occupation score and thus vehicle reduction than the greedyShared algorithm. In fact with an increase of the time score of 30% with the threshold, it can achieve around 30% vehicle reduction compared to greedyShared algorithm on the smaller scale network. In this case, the computation time of the method is largely higher than of greedyShared but it is still realistic in practical applications in urban transportation systems. The biggest drawback of this method is that it cannot handle big network and this could be a direction for improvement of the method.

These findings suggest that multi-objective threshold simulated annealing approach is effective in optimizing both fleet size and service efficiency, providing a

promising direction for further research.

# Chapter 6

# Discussion

## 6.1. Computation Time

As seen earlier, the computation time [22] of the proposed method is still less efficient than the greedyShared in large road networks, yielding worse results. However, the results can be better on smaller-scale networks. A way to tackle this issue is to make the computation more scalable [23].

In the experimentation, a very large network with many vehicles was used. A potential improvement could be to use a grid and process simulations on smaller parts of the map for trips that stay within one cell of the grid. For trips that cross cells, cells can be merged to create larger maps with fewer vehicles.

An other improvement would be to do paralleled computing to try several neighbors at the same time with different machines instead of one at a time. This would permit to do Simulated Annealing exploration phase faster.

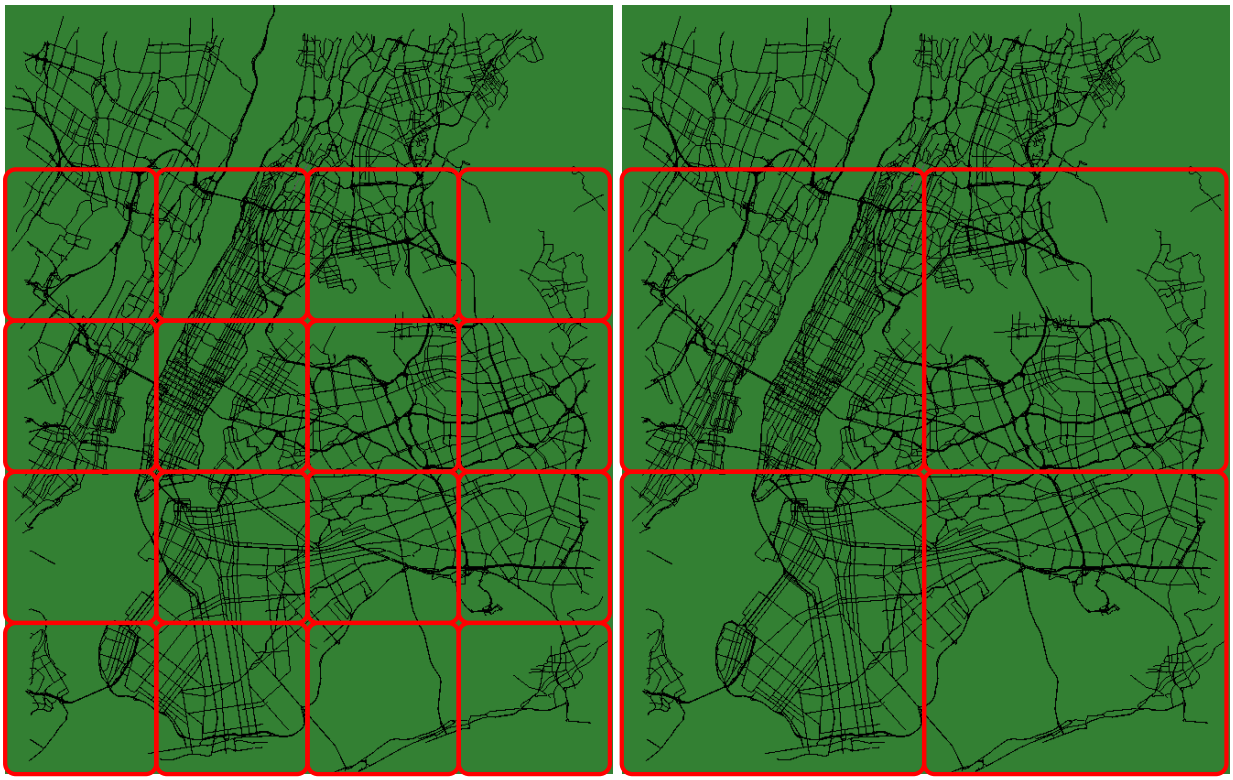

Figure 13: On the left side the small grid and on the right side the merged cell grid for trip across these merge cells

# Chapter 7

# Conclusion

This master thesis, explored the optimization of urban taxi systems using a modified version of simulated annealing. The primary objective was to reduce the number of vehicles required while maintaining high levels of service efficiency. The research leveraged real-world taxi data from New York City to validate the proposed optimization model.

The proposed method effectively balances the trade-off between minimizing fleet size and ensuring efficient service. The integration of multi-objective optimization into the dispatching framework addressed the limitations of traditional single-objective approaches and algorithms that prioritize immediate efficiency.

Through detailed simulations and comparisons with existing methods, the proposed approach demonstrated significant improvements. Specifically, it achieved a reduction in the number of vehicles needed without compromising overall service quality. These findings highlight the potential of optimized autonomous taxi systems to enhance urban mobility, reduce congestion, and contribute to more sustainable transportation solutions.

Though the computation time is not yet at a level of a practical application in urban transportation systems and need improvement on large road networks with some different techniques such as those explained in the discussion chapter.

Future research could further refine the optimization model by incorporating additional real-world constraints and exploring other advanced optimization tech-

niques. The scalability of the proposed method to larger datasets and different urban environments also warrants further investigation.

# References


[1] Emily Nagler. Standing still, July 2021.

[2] Bob Pishue. Inrix 2022 global traffic scorecard. https://inrix.com/scorecard/, 2023. Accessed: 2024-07-08.

[3] Taxi and Limousine Commission (TLC). 2015 green taxi trip data. https://data.cityofnewyork.us/Transportation/2015-Green-Taxi-Trip-Data/gi8d-wdg5/about_data, 2015. Accessed: 2024-07-08.

[4] Pengzhan Zhou, Cong Wang, and Yuanyuan Yang. Design and optimization of solar-powered shared electric autonomous vehicle system for smart cities. 22(4):2053–2068. Conference Name: IEEE Transactions on Mobile Computing.

[5] Menghai Pan, Yanhua Li, Zhi-Li Zhang, and Jun Luo. SCCS: Smart cloud commuting system with shared autonomous vehicles. 8(5):1301–1311. Conference Name: IEEE Transactions on Big Data.

[6] Yanchang Liang, Zhaohao Ding, Tao Ding, and Wei-Jen Lee. Mobility-aware charging scheduling for shared on-demand electric vehicle fleet using deep reinforcement learning. 12(2):1380–1393. Conference Name: IEEE Transactions on Smart Grid.

[7] Tingwen Zheng and Yanping Jiang. Driver-rider matching and route optimization in carpooling service for delivering intercity commuters to the high-speed railway station. 227:120231–. Publisher: Elsevier Ltd.

[8] Sondes Ben Cheikh-Graiet, Mariagrazia Dotoli, and Slim Hammadi. A tabu search based metaheuristic for dynamic carpooling optimization. *Computers Industrial Engineering*, 140:106217, 2020.

[9] Sheng Su, Dongwen Xiong, Haijie Yu, and Xiaohua Dong. A multiple leaders particle swarm optimization algorithm with variable neighborhood search for multiobjective fixed crowd carpooling problem. *Swarm and Evolutionary Computation*, 72:101103, 2022.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[11] Michael Behrisch Daniel Krajzewicz, Jakob Erdmann and Laura Bieker. Recent development and applications of sumo – simulation of urban mobility. *International Journal on Advances in Systems and Measurements*, 5(3-4):128–138, 2012.

[12] Jakob Erdmann. MSDispatch_GreedyShared.cpp. `https://github.com/eclipse-sumo/sumo/blob/main/src/microsim/devices/MSDispatch_GreedyShared.cpp`.

[13] SUMO User Conference 2021. Simulation of ride-pooling using drtonline and greedyshared algorithms. In *SUMO User Conference 2021*, 2021.

[14] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[15] Paolo Serafini. A multiobjective optimization method based on simulated annealing. In *Proceedings of the International Conference on Operations Research*, pages 61–70, 1994.

[16] New York City Taxi and Limousine Commission. Green cabs, 2024. Accessed: 2024-08-26.

[17] Sodapy Contributors. Sodapy, 2024. Accessed: 2024-08-26.

[18] OData. Odata: Open data protocol, 2024. Accessed: 2024-08-26.

[19] SUMO Team. Traci: Traffic control interface, 2024. Accessed: 2024-08-26.

[20] SUMO Team. Osmwebwizard, 2024. Accessed: 2024-08-26.

[21] SUMO Team. netgenerate, 2024. Accessed: 2024-08-26.

[22] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. Complexity and approximation: Combinatorial optimization problems and their approximability properties. *Journal of Complexity*, 6(1):53–78, 1980.

[23] Ian Sommerville. Distributed software engineering. In *Software Engineering*, chapter 18. Addison-Wesley, Boston, 9th edition, 2011. Discusses aspects related to distributed systems and scalability.

# Acknowledgements

This thesis was carried out under the direction and guidance of Professor Naoaki Yamanaka and Professor Hiroshi Shigeno of Keio University, whose insights were invaluable throughout the process. I am deeply grateful to my family for their formidable financial support, sustaining my living expenses during this journey. Lastly, I would like to extend my thanks to the Keio buildings. It provided me with targets when I did not want to leave my home, nourished my curiosity and experimental mind, and supplied materials, hardware, and unidentified goods. It played a crucial role in helping me maintain my mental sanity.

*School of Science for Open and Environmental Systems*
*Graduate School of Science and Technology*
*Keio University*

July 2024
Glenn LOUÉDEC