Degree project in Civil and Architectural Engineering

First cycle, 15 credits

Modeling Autonomous On-Demand Public Transport

Churong chen

Abstract

As autonomous vehicle (AV) technology evolves and matures, automated public transit (APT) is gaining attention due to its flexibility, cost-effectiveness, and efficiency. This report explores various algorithms for allocating vehicles to passengers within APT systems. It aims to organize and propose effective allocation strategies and validate them through comparative analyses on test networks. Overall, the paper introduces several algorithms, with six specifically compiled and tested using the VIPSim simulator across four traffic networks. Two of these networks are basic, while the other two are more complex and represent real-world scenarios. Through these numerical experiments, the algorithm that maximizes network operational efficiency was identified, and several instructive conclusions were drawn from the comparative analysis.

Keywords

Automated Public Transit, autonomous vehicle, assignment methods, ride-sharing, simulation

Sammanfattning

När tekniken för autonoma fordon (AV) utvecklas och mognar, får automatiserad kollektivtrafik (APT) uppmärksamhet på grund av sin flexibilitet, kostnadseffektivitet och effektivitet. Denna rapport utforskar olika algoritmer för att tilldela fordon till passagerare inom APT-system. Den syftar till att organisera och föreslå effektiva allokeringsstrategier och validera dem genom jämförande analyser på testnätverk. Sammantaget introducerar rapporten flera algoritmer, varav sex specifikt har sammanställts och testats med hjälp av VIPSim-simulatorn över fyra trafiknätverk. Två av dessa nätverk är grundläggande, medan de andra två är mer komplexa och representerar scenarier från verkliga världen. Genom dessa numeriska experiment identifierades algoritmen som maximerar nätverkets operativa effektivitet, och flera instruktiva slutsatser drogs från den jämförande analysen.

Nyckelord

Automatiserad kollektivtrafik, autonoma fordon, tilldelningsmetoder, samåkning, simulering

Acknowledgments

Spending my final semester at the Royal Institute of Technology (KTH) in Sweden for a semester exchange was truly magical and undoubtedly the best decision I have ever made. I had an incredibly rewarding and joyful semester at KTH! I must say, I fell in love with Stockholm—the people, the learning environment, the weather, and the air here are all phenomenal. I am deeply grateful to my undergraduate institution and KTH for this opportunity, which has deepened my reflections and shown me many possibilities beyond my home country, moving me profoundly.

I extend my heartfelt thanks to my degree project supervisor, Dr. Wilco Burghout, who guided me through the concepts of Automated Public Transit, encouraged my critical thinking, and assisted me in compiling algorithms for the simulations. I feel extraordinarily fortunate to have encountered such a responsible and kind-hearted professor in a foreign land. I am immensely thankful that Dr. Wilco agreed to take me under his guidance.

I would also like to express my gratitude to Dr. Albania Nissan for her overarching guidance. I particularly enjoyed the atmosphere of the thesis defense preparation, which was both relaxing and enlightening.

Additionally, I thank ChatGPT, for its help in refining the English in my writing and correcting grammatical and lexical errors.

Contents

Abstract....2

Keywords....2

Sammanfattning....3

Nyckelord....3

Acknowledgments....4

Contents....5

1 Introduction....7

1.1 Background....7

1.2 Research Questions....8

1.3 Delimitations....8

1.4 Disposition....9

2 Related work....10

2.1 Autonomous Vehicles....10

2.2 Autonomous Public Transport....11

2.3 Assignment Methods....11

2.4 Ride-Sharing Methods....13

3 Problem Statement....15

3.1 Problem Settings....15

3.2 Mathematical Model....16

3.3 Simulation Networks....17

4 Methodology....21

4.1 Simple nearest neighbour assignment....21

4.2 Dynamic Assignment....26

4.3 Ride-Sharing Assignment....29

4.4 Algorithms to be tested....33

5 Numerical Experiment....38

5.1 Basic Algorithm....38

5.2 Enhanced Algorithm 40
5.3 Advanced Algorithm 43
5.4 Overall Test Results 45
6 Conclusion 49
6.1 Summary 49
6.2 Future Direction 49
6.3 Sustainability 50
Reference 52
Appendix 55

1 Introduction

1.1 Background

Autonomous driving has rapidly become a pivotal topic within the realm of transportation technology, evolving significantly in recent years. This evolution marks a monumental stride in transportation innovation, potentially reshaping the landscape of mobility and efficiency(Clavijo, Jiménez and Naranjo, 2023). The primary benefits of autonomous driving extend well beyond relieving humans from the task of driving; they encompass the capability of vehicles to collect and analyze varying spatiotemporal data, thereby facilitating more informed and efficient driving decisions and route planning. Consequently, autonomous vehicles (AVs) are poised to become a fundamental component of smart city infrastructures.

While the organization and design of public transportation are crucial components of large cities, aiming to enhance sustainability and urban living standards, there are still various issues that persist(Iclodean, Cordos and Varga, 2020). Despite each city facing unique challenges, the general goals remain consistent: to improve the operational efficiency of public transportation, increase the passenger capacity per bus, reduce passenger waiting times, and decrease the operating costs associated with these systems, among others. Therefore, integrating autonomous driving with public transport and incorporating this synergy into urban settings can offer numerous benefits.

In this report, we propose the replacement of traditional buses with autonomous buses to improve the public transportation system. Specifically, rather than operating buses on fixed routes, we suggest assigning buses to passengers based on demand. This approach would utilize real-time data on passengers and vehicles on roads to intelligently assign vehicles to passengers, thereby achieving optimal objectives. Such a system offers multiple advantages. On-demand transportation enables dynamic rapid transit, enhancing the flexibility of public transport and allowing for the expansion of various features, such as enabling passengers to reserve seats through an app, thereby improving communication between passengers and the public transport system. Additionally, in suburban areas, where demand is more sporadic, supplying vehicles on-demand can significantly reduce the occurrence of empty runs. Furthermore, this approach provides efficient first- and last-mile connections, particularly to areas with low demand(Carrese *et al.*, 2023). From an economic perspective, incorporating AVs into the public transport system can also be beneficial. By combining AVs with existing public transport frameworks, there's a substantial opportunity to alleviate traffic congestion significantly, thus contributing to a more sustainable and efficient urban transit system(Poinsignon *et al.*, 2022).

There are notable examples of practical initiatives where AVs have been integrated into existing public transportation networks. In Rouen, France, for example, a fleet of four AVs has been seamlessly integrated, demonstrating the critical role of collaboration and regulatory frameworks in adapting these vehicles to meet local transit needs. Similar initiatives have been launched in Sitten, Switzerland, where autonomous buses have been operational since 2016, and in Lyon, France, and Michigan, USA. Collectively, these autonomous buses have accumulated over 50,000 kilometers in travel distance and have transported more than 100,000 passengers(Pakusch and Bossauer, 2017).

1.2 Research Questions

However, there are several critical issues still to be addressed in the field of Autonomous Public Transport (APT), such as how to assign vehicles to passengers. Furthermore, the criteria for assigning vehicles to passengers need to be established: should the aim be to maximize the number of people transported, minimize passengers' waiting time, or provide the fairest public transport (PT) service? What impacts might focusing on a single objective have on other objectives? How can we compromise between different objectives? These are all questions that require further analysis and research. Therefore, in this report, we aim to discuss different assignment algorithms and do simulation tests on them. Thus, reaching constructive conclusions.

1.3 Delimitations

At the same time, there are some issues that this report will not discuss. For example, the report will not discuss the scope of APT implementation, nor will it address the pricing issues of such APT systems, nor the safety concerns of AVs. Even though these issues are practically significant in reality, we will focus on the main topic of our research. Additionally, our test network is within a certain range, such as a part of Goteborg.

1.4 Disposition

In the following report, related work in AVs and APT assignment algorithms will be presented in Chapter 2. In Chapter 3, the settings for APT, the mathematical models used, and the simulation network will be introduced. Chapter 4 will introduce the algorithms we propose. Chapter 5 will display the test results of the algorithms on the simulation network. Finally, the conclusion will be discussed in Chapter 6.

2 Related work

2.1 Autonomous Vehicles

The concept of AVs was introduced over 30 years ago, transitioning from the realm of science fiction to a scientific reality. This shift has been fueled by rapid advancements in information technology, artificial intelligence, and wireless communications, particularly vehicle networking and V2X communications. AVs are equipped with numerous sensors, including rangefinders, radars, cameras, GPS modules, and gyroscopes. These sensors provide comprehensive perception capabilities that enable the vehicle to adapt to its surroundings(Lam, Leung and Chu, 2016). Additionally, AVs can establish informational connections with other vehicles and infrastructure, facilitating better decision-making. A significant number of companies are investing in autonomous driving technologies and planning to produce AVs. For instance, Google conducted tests on driverless cars without a safety driver in 2011. Companies like Waymo, Tesla, and Ford are also developing driverless vehicles. Research consistently indicates that AVs will soon become mainstream, fundamentally transforming human mobility.

There has been extensive research on AVs, primarily focusing on technology, safety, and liability(Tafidis *et al.*, 2022), as well as travel behavior(Soteropoulos, Berger and Ciari, 2019), land use(Soteropoulos, Berger and Ciari, 2019), and human-autonomy (H-A) collaboration(Xing *et al.*, 2021). These studies explore various dimensions of how AVs impact our lives, from technical advancements and the responsibility issues they entail to their effects on travel patterns and urban planning and even the collaborative dynamics between humans and autonomous systems.

The introduction of AVs has brought numerous benefits to urban transportation. With rapid urban population growth, the concept of the "smart city" is increasingly discussed. A critical aspect of the smart city is "smart transport," where existing literature extensively refers to AVs as a solution to meet the escalating demands of urban transportation, aiming for greater efficiency, safety, and sustainability. Studies suggest that AVs enhance traveler convenience and safety, reduce the negative impacts of congestion, and are more environmentally friendly(Litman, 2014). Additionally, it is proposed that AVs can intelligently adjust the timing, speed, distance, and other driving behaviors between vehicles(Kyriakidis, Happee and De Winter, 2015). Vehicles use various on-board wireless communication technologies for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure communications, expanding the possibilities for AVs. Research focusing on the wireless communication framework for driverless city vehicles has demonstrated improvements in safety and efficiency through enhanced communications(Furda *et al.*, 2010). Moreover, optimization models for traffic light signals to vehicles (TLS2V) and V2V communications have been developed to reduce fuel usage and emissions(Alsabaan, Naik and Khalifa, 2013).

2.2 Autonomous Public Transport

Public transportation is a vital component of urban mobility, generally categorized into two main types: rail and road. Rail-based public transport systems rely on a network of tracks, while road public transport primarily includes buses and taxis. Buses have a larger passenger capacity and can serve more passengers per trip. However, they are less flexible, operating on fixed routes and schedules. Taxis offer high flexibility, with routes that can change in real-time based on passenger demand, but they have a smaller capacity. The question arises whether there is a mode of public transport that could integrate the advantages of both buses and taxis. This would involve offering on-demand services to maintain flexibility while supporting ride-sharing to enhance efficiency(Lam, Leung and Chu, 2016). APT systems are proposed as a solution to achieve this goal. Such public transport methods are beneficial; research shows that adding a sharing feature to taxis can decrease total travel distance by 40% or more, despite a slight increase in passenger discomfort. These benefits come with reduced service costs, emissions, and staggered fares, suggesting wide passenger acceptance of this shared service(Santi *et al.*, 2014). This also highlights the promising prospects of APT.

2.3 Assignment Methods

In the operation of APT within urban areas, two primary considerations are pivotal: (1) strategies for assigning AVs to traveler demand requests, and (2) sharing strategies for AVs.

For the assignment of AVs to travelers, a common strategy is the heuristic approach.

The most basic heuristic method focuses on minimizing the longest waiting time, utilizing the first-come, first-served (FCFS) principle to assign travelers to the nearest available AV. This helps in reducing wait times by efficiently allocating nearby AVs as demand arises(Jordan, Llc and Scarborough, 2013). In addition, the FCFS method is also employed by Zhang and others, where users calling simultaneously are randomly sorted. This randomization prevents the system from consistently prioritizing the same areas, ensuring a more equitable service distribution across different urban regions(Zhang *et al.*, 2015). Furthermore, in 2016, Chen and colleagues improved this approach by refining the search for the "nearest" available AV. They segmented the overall area into sub-regions, initially searching for an available AV within the same sub-region, and expanding the search to adjacent sub-regions if necessary. This method helps in efficiently managing the distribution of AVs and reducing the time it takes for an AV to reach a requester(Chen, Kockelman and Hanna, 2016). Differing from the traditional priority of servicing the traveler with the longest waiting time, another approach focuses on minimizing the number of vehicles dispatched. This involves incorporating passengers en route into ongoing trips, even if this deviates from the FCFS principle. This adjustment is permitted because it does not add any extra travel time for the AVs, thereby optimizing the use of vehicles and minimizing unnecessary travel(Levin *et al.*, 2017).

Additionally, there are optimization-based methods for assigning AVs. An approach utilizes optimization strategies to dynamically assign AVs to traveler requests, allowing for the reassignment of previously assigned travelers to other AVs as new requests enter the system. This method aims to minimize the total waiting time for travelers or the distance traveled by vehicles(Hyland and Mahmassani, 2018). One study developed a mixed integer linear programming model to tackle the single-household shared AV problem(Cokyasar and Larson, 2020).

Another approach employed the maximum weight bipartite matching model to maximize system-wide travel savings in a dynamic ride-sharing context. By constructing a bipartite graph, each driver and passenger is represented as a node, and feasible pairings are depicted as edges. The weight of each edge represents the travel distance saved by the pairing. This method also considers the matching needs of round-trip journeys by adding constraints to ensure simultaneous matching for these trips(Agatz *et al.*, 2011). Furthermore, a game-theoretical framework has been applied to the assignment of AVs to passengers. This method structures the vehicle-target assignment problem as a multiplayer game and uses pure Nash equilibrium to represent equitable vehicle assignments, aiming to maximize overall system utility(Arslan, Marden and Shamma, 2007).

2.4 Ride-Sharing Methods

In the study of ride-sharing strategies for AVs, a classical method involves the use of shareability networks. Santi and colleagues introduced such a graph model where each node represents an individual trip, and the edges between nodes indicate that these trips can be reasonably shared. This network is transformed into a graph matching problem, utilizing maximum matching or maximum weight matching algorithms to find optimal trip pairings, thereby maximizing the number of shared trips or minimizing the total time required to complete all trips(Santi *et al.*, 2014).

Further extensions of this method have been applied to the sharing of a large number of passengers and trips. Improvements include adding vehicles to the shareability network and constructing an integer linear optimization model on this network to enable both trip-to-trip and vehicle-to-trip matching(Alonso-Mora *et al.*, 2017). Another approach adapts this model to allow a single vehicle to match with only one new request while still handling multiple requests simultaneously. This method involves transforming the carpooling problem into a joint optimization framework involving linear assignment between fleet vehicles and customer trip requests, which results in a significantly faster algorithm(Simonetto, Monteil and Gambella, 2019). Additional refinements involve addressing one-to-one ride-matching problems by breaking down the original graph into multiple subgraphs and introducing various clustering algorithms to boost computational efficiency, optimizing the process for dynamic ride-sharing scenarios(Tafreshian and Masoud, 2020).

However, there remain many areas in previous research that have not been thoroughly explored. For example, some algorithms have not been tested in micro-simulation environments, or, in other words, lack randomness, making them unable to accurately represent real-world traffic conditions. Moreover, while many assignment algorithms reference FCFS and introduce new, more efficient algorithms based on it, they do not clearly indicate how they balance efficiency and fairness. Additionally, the applicability of these algorithms may be related to the characteristics of the network and the scale of the demand, which has been rarely mentioned in prior literature. The availability of historical data and real-time mobile data makes it possible to predict passenger behavior, and proactive vehicle assignment based on predictions could be highly beneficial in APT, but few studies have considered this aspect.

Since many studies have not used simulations that accurately represent real traffic conditions to test their algorithms, and because each algorithm has been tested on different networks and environments without a unified testing platform, this report aims to use simulation to test, investigate, and analyze these algorithms. In this report, we use four different testing networks, starting with two simple networks (one is a rhombus network with equal sides, and the other is a quadrilateral network with unequal sides), and then using more complex real networks to test different algorithms. We will discuss and analyze whether the characteristics of the network impact the efficiency of the algorithms and their comparative performance.

3 Problem Statement

In this section, we will present an introduction to the settings of our APT system, the mathematical model of the system, and the simulated networks on which we will conduct our numerical experiments.

3.1 Problem Settings

In the APT system, we aim to study, autonomous buses that transport passengers within a designated area. This area contains several predefined stations, and the autonomous buses operate between these fixed stations, as can be seen in Figure 1. Passengers need to wait at the stations, and since this study does not consider features such as passenger bookings through an app, autonomous buses can only obtain passenger travel information when passengers arrive at the station. The buses then adjust their routes accordingly. The primary focus of this study is how these autonomous buses can be optimally assigned to passengers based on the available information. Additionally, autonomous buses also consider ride-sharing to maximize the utilization of their large capacity. The capacity per vehicle adopted in this article is 10. The key difference between autonomous buses and regular public transportation buses is that regular buses run on fixed routes and schedules, while autonomous buses are assigned on-demand based on passenger needs, dynamically adjusting their routes in real-time. Figure 1 is an illustration of the APT system.

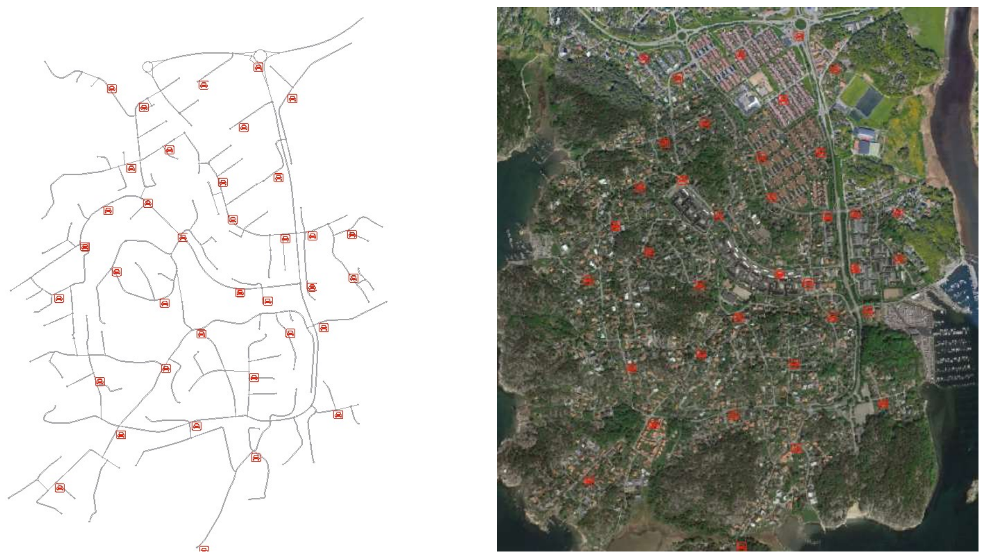


Figure 1. Illustration of APT system.

3.2 Mathematical Model

The network will be represented by a graph $G = G(V, A)$, where every vertex $v \in V$ is an intersection or a bus station, and every segment $a \in A$ represents a section of the network. Each segment a contains the information of the starting point, the terminal point, and the time required to traverse the segment, which can also be considered as the segment's distance.

The demand is represented by the arrival of the passengers p. In this report, two types of demand are considered: (1) The first type consists of sequentially arriving passengers, for whom we know the station of arrival, the time of arrival, and their desired destination. (2) The second type of demand is "bunch demand," where the information provided includes the station of arrival, the time of arrival, the total number of passengers, and their respective desired destinations. The reason for considering such a "bunch arrival" is that our APT system is designed to integrate with other modes of transportation, thereby forming a multi-modal transport system. For example, the arrival of trains or subways might bring a group of passengers to a station.

There is also a vehicle set to indicate the locations of the vehicles, their destinations, the number of passengers on board, and whether the vehicle is already assigned to some certain passengers.

The inputs of the model are: (1) the graph $G = G(V, A)$; and (2) the demands, which contains information such as the passengers' arrival time at the station, the number of passengers, and their destinations. The output of the model is the set of all movements performed by the vehicles.

There are many potential objective functions for evaluating or optimizing the aforementioned APT system. One could aim to minimize the maximum waiting time or minimize the average waiting time. These two different objectives address individual goals and network-wide goals, respectively. When minimizing the maximum waiting time, the overall network efficiency might be compromised to prioritize passengers with longer waiting times. On the other hand, when minimizing the average waiting time, some passengers in remote areas might never be served. Alternatively, from the perspective of vehicle operation, the objective function could be to maximize the total number of passengers served within a certain time or the ratio of effective trips (assuming trips with passengers on board are considered effective). Different objective functions will lead to different evaluations of the system and result in different optimization algorithms.

3.3 Simulation Networks

In our report, we will utilize VIPSim, a micro-level traffic simulation software, as our primary simulation tool. This software allows for the importation of specific maps to set up designated stations, predefine passenger travel patterns, and guide the movement trajectories of autonomous buses using various methods, including an element of randomness to better reflect real-world scenarios.

The study will be conducted across four different traffic networks to test various proposed algorithms—from the simplest networks to more complex ones representing portions of real urban areas—aiming for a comprehensive analysis of the algorithms ' performance across network types.

Simulation case 1: The first traffic network is designed as a simple square grid comprising four stations, as shown in Figure 2, and we will refer to it as "simplesquare" in the following text. Vehicles move along diagonal roads, resulting in equal travel distances between each station and the remaining three. This design significantly simplifies the road network complexity, but it may also introduce certain considerations. For instance, the difference in dispatching an idle bus from the north station to the south versus to the west is not markedly distinct. Consequently, this could obscure certain factors during algorithm analysis.

The demand used in this network includes both individual demand and bunch demand. The bunch demand takes into account situations when a large public transport vehicle, such as a train, arrives at the station, and a group of passengers arrives at the same time. The total demand in the simulation is 299 passengers. Additionally, there are a total of 3 AVs in this network. And the simulation time is 7200 simulation steps, which is 2 hours in total.

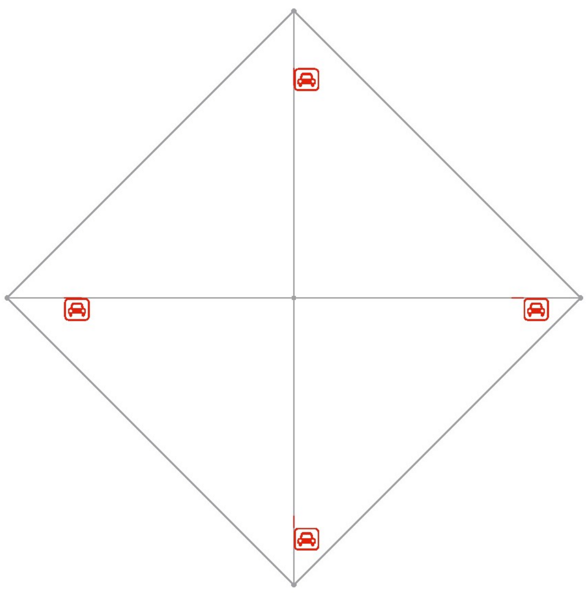


Figure 2. "Simplesquare" network.

Simulation case 2: The second traffic network, is shown in Figure 3, which will refer to it as "simplesquare (2)", introduces additional complexity to the initial grid by extending the station on the right further outward. This change disrupts the symmetry among the stations, effectively addressing the considerations previously mentioned. By integrating this modified layout with the initial traffic network, we enhance our capacity to conduct a more precise analysis of algorithm performance across simpler networks. This simulation network shares the same demand, simulation time with "simplesquare" network.

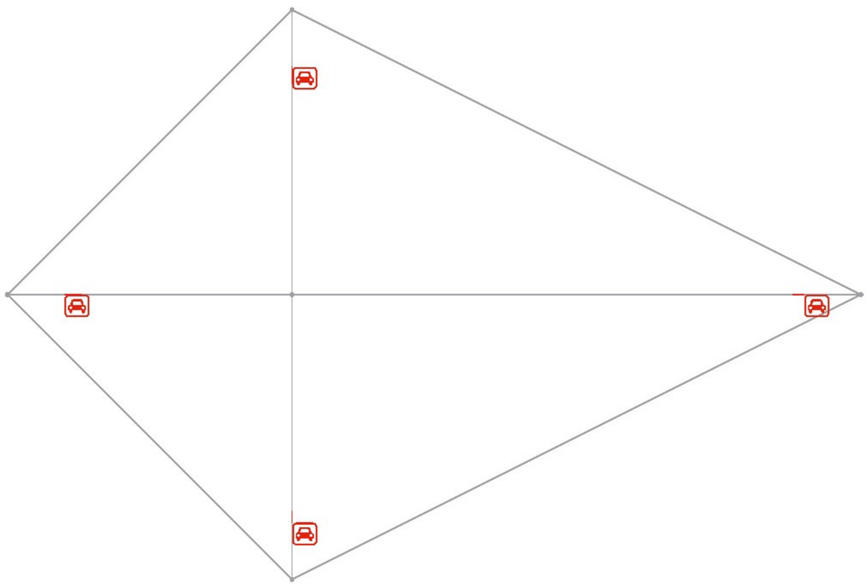


Figure 3. "Simplesquare (2)" network.

The previously discussed networks are simplistic road networks, which enable us to analyze each specific vehicle movement in detail. However, they clearly do not represent the conditions experienced in the real world. To examine the effects of algorithms on a more macroscopic level across an entire region, it is necessary to consider larger networks. Therefore, we have incorporated two networks that exist in the real world.

Simulation case 3: The "Saclay" network, located in Saclay, France, consists of 22 stations and is characterized by its considerable length in the longitudinal direction and shorter width in the transverse direction. This configuration provides a unique framework for studying traffic flow dynamics and algorithm efficiency in a longitudinally extended urban setting. Figure 4 shows the network in the simulator, while Figure 5 shows the network in the map.

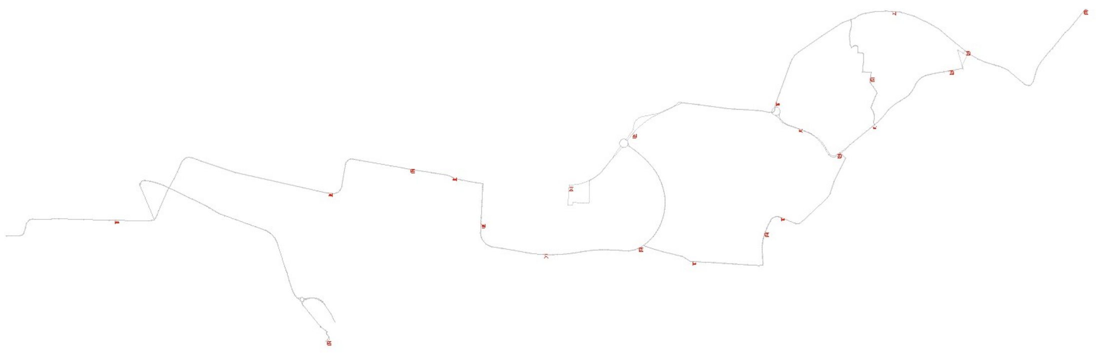


Figure 4. "Saclay" network (a).

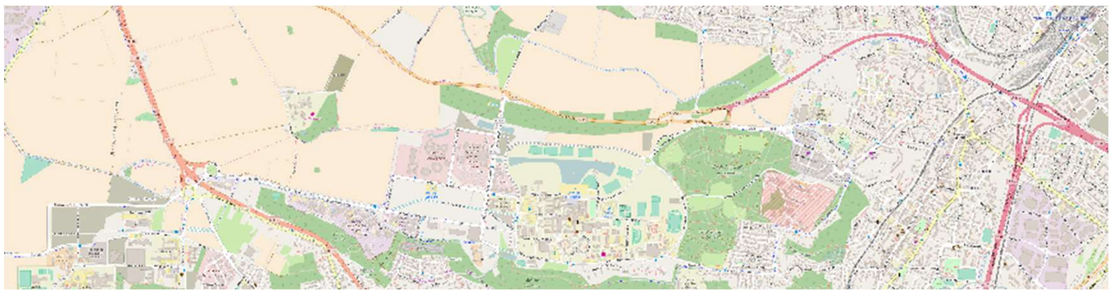


Figure 5. "Saclay" network (b).

The "Saclay" network also contains individual demand and bunch demand, with a total demand of 560 passengers. The simulation time is 3600 simulation steps, equivalent to one hour.

Simulation case 4: Subsequently, we introduced a part of the road network located in the southwest corner of Goteborg, as shown in Figure 6. Compared to the "Saclay" network, the "Goteborg" network contains more stations and exhibits a more complex structure. It distinctly presents a grid-like configuration. A notable feature of the "Goteborg" network, as shown in Figure 6, is a station situated in the northeast corner of the considered area, at the upper right of the image. This station acts as a crucial transport hub connecting the southwest area of Goteborg with other parts. The deployment of APT in this region aims to provide on-demand public transport services in areas with sparse demand. This setup not only meets the local mobility needs within the region but also supports extensive travel to the transport hub for reaching other areas.

Considering the actual circumstances, unlike the three networks mentioned above, this network only includes individual demand, with a total demand of 591 passengers. The simulation time is one hour. In this example, the efficiency of the algorithm can be tested, demonstrating the significant role and practical applications of APT in urban transportation.

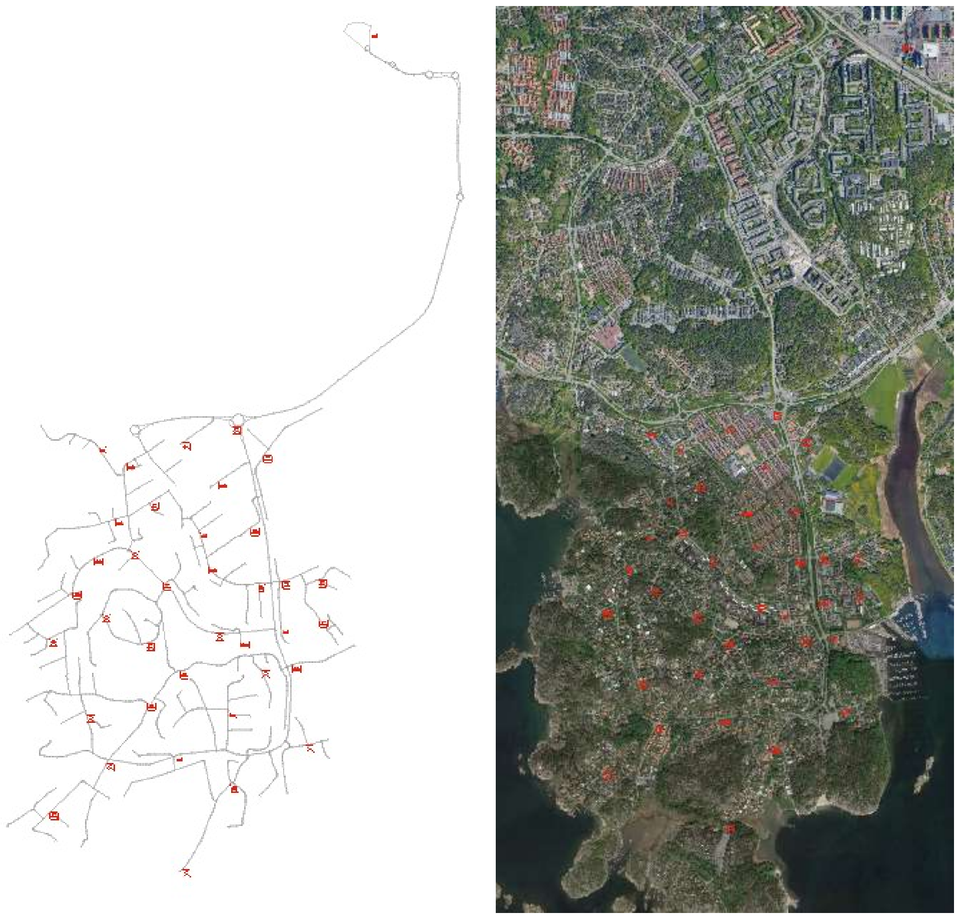


Figure 6. "Goteborg" network.

4 Methodology

In this section, we will provide a detailed explanation of the algorithms considered for allocating autonomous buses to passengers. We begin with the basic algorithm based on the simple nearest neighbour assignment and then progress to more sophisticated algorithms that incorporate ride-sharing during the assigning process.

4.1 Simple nearest neighbour assignment

The Simple Nearest Neighbor (SNN) assignment algorithm strategically assigns the closest available buses to passengers based on specific criteria. Illustrated in Figure 7 is an operational diagram of this algorithm, where assignments are primarily based on the length of time passengers have waited at the station, adhering to the FCFS principle. This criterion can be adjusted to account for either the total number of people waiting at the station or the count of unassigned passengers. One of our key objectives is to evaluate the operational efficiency of algorithms under these varying criteria. Notably, the "Find ride-share candidates" step highlighted in Figure 7 refers to a ride-sharing algorithm that functions independently from the assignment mechanism. This feature enables passengers to board a bus if both the start and end points of the bus's route align with their own destinations, provided that the bus has available seats. This ride-sharing algorithm will be integrated into all subsequent algorithms discussed in our study.

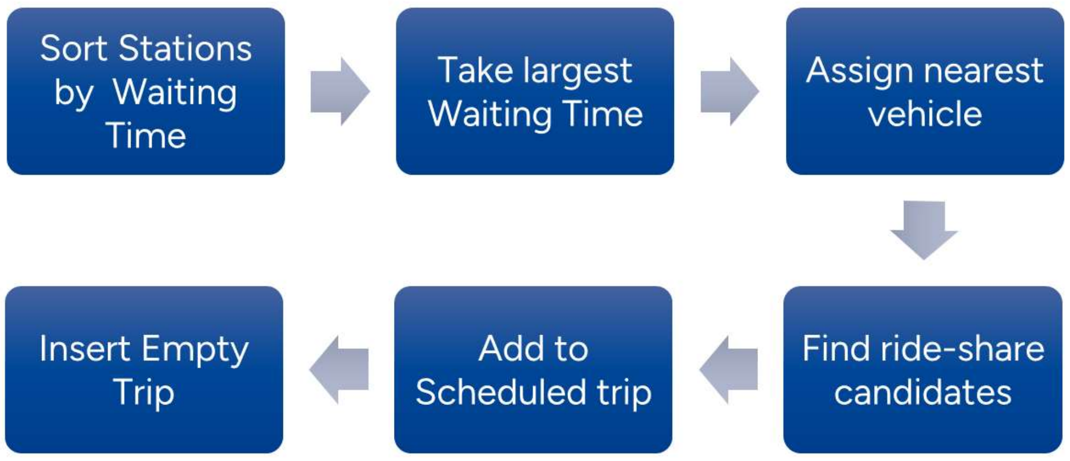

Sort Stations by Waiting Time
Take largest Waiting Time
Assign nearest vehicle
Find ride-share candidates
Add to Scheduled trip
Insert Empty Trip

Figure 7. Operational diagram of SNN assignment algorithm.

The pseudocode for the algorithm is as follows:

Algorithm 1 SNN (MaxWaitSimple)

```
Input: Graph $G = G(V, A)$;
        Predefined demands: number of passengers $n_p$, arriving time $t_p$;
        Vehicle set;
Output: All movements.
For every 20 simulation steps:
        SortStation ← sort by the longest waiting time of unassigned passengers.
        $s$ ← top (SortStation).
        $p$ ← longest waiting unassigned passenger at station $s$.
        If station $s$ has empty vehicle $v$ then:
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$.
                Chain $trip_p$ to Movements.
                Remove $p$ from s.unassigned_passengers.
        Else:
                Find nearest empty vehicle $v$.
                Create EmptyTrip $emt_p$.
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$ and $emt_p$.
                Chain $emt_p$ to Movements.
                Chain $trip_p$ to Movements.
                Remove $p$ from s.unassigned_passengers.
        EndIf
End
```

If the criterion is changed to the number of unassigned passengers at the station, the pseudocode remains similar and is presented as follows:

Algorithm 2 SNN (MaxNrPassengers)

```
Input: Graph $G = G(V, A)$;
        Predefined demands: number of passengers $n_p$, arriving time $t_p$;
        Vehicle set;
Output: All movements.
For every 20 simulation steps:
        SortStation ← sort by the number of unassigned passengers.
        $s$ ← top (SortStation).
        $p$ ← longest waiting unassigned passenger at station $s$.
        **If** station $s$ has empty vehicle $v$ **then:**
            Create Trip $trip_p$.
            Assign $v$ and $p$ to $trip_p$.
            Chain $trip_p$ to Movements.
            Remove $p$ from s.unassigned_passengers.
        **Else:**
            Find nearest empty vehicle $v$.
            Create EmptyTrip $emt_p$.
            Create Trip $trip_p$.
            Assign $v$ and $p$ to $trip_p$ and $emt_p$.
            Chain $emt_p$ to Movements.
            Chain $trip_p$ to Movements.
            Remove $p$ from s.unassigned_passengers.
        **EndIf**
    **End**
```

Considering the large number of passengers and vehicles, assigning only once every 20 simulation steps (where 20 serves as a hyperparameter that can be adjusted according to specific needs) might be inadequate. Thus, it is proposed to continue the assignment process until there are no remaining vehicles or unassigned passengers. The pseudocode for this refined approach is outlined below:

Algorithm 3

```
**Input:** Graph $G = G(V, A)$;
        Predefined demands: number of passengers $n_p$, arriving time $t_p$;
        Vehicle set;
**Output:** All movements.
**For** every 20 simulation steps:
    SortStation ← sort by the longest waiting time of unassigned passengers.
    **While** the number of unassigned passengers are not zero **do**:
        $s$ ← top (SortStation).
        $p$ ← longest waiting unassigned passenger at station $s$.
        **If** station $s$ has empty vehicle $v$ **then:**
```

```
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$.
                Chain $trip_p$ to Movements.
                Remove $p$ from s.unassigned_passengers.
            Else:
                Find nearest empty vehicle $v$.
                Create EmptyTrip $emt_p$.
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$ and $emt_p$.
                Chain $emt_p$ to Movements.
                Chain $trip_p$ to Movements.
                Remove $p$ from s.unassigned_passengers.
            EndIf
        EndWhile
End
```

Previous algorithms considered the assignment of empty vehicles exclusively. However, consider a scenario where all buses in a current area are too far from the passenger p, who has been waiting the longest. Meanwhile, some vehicles, although currently carrying passengers, are soon to arrive at the station where passenger p is standing to drop off passengers and become available. Subsequently, these vehicles can transport passenger p to their destination. Including such incoming buses in the assignment algorithm can significantly reduce the waiting time for the passenger who has waited the longest, better meeting the FCFS criteria by actually transporting passengers (rather than merely assigning them based on FCFS). This approach also avoids the inefficiency of distant empty vehicles traveling to pick up passenger p, thereby making the assignment process both fairer and more efficient. An improved pseudocode based on Algorithm 1 is outlined as follows:

Algorithm 4

```
Input: Graph $G = G(V, A)$;
       Predefined demands: number of passengers $n_p$, arriving time $t_p$,
       destination $s_p$;
       Vehicle set;
Output: All movements.
```

```
For every 20 simulation steps:
    SortStation ← sort by the longest waiting time of unassigned passengers.
    $s$ ← top (SortStation).
    $p$ ← longest waiting unassigned passenger at station $s$.
    If station $s$ has empty vehicle $v$ then:
        Create Trip $trip_p$.
        Assign $v$ and $p$ to $trip_p$.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
    Else If station $s$ has incoming vehicle $v$' that will be empty:
        Create Trip $trip_p$.
        Assign $v'$ and $p$ to $trip_p$.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
    Else:
        Find nearest empty vehicle $v$.
        Create EmptyTrip $emt_p$.
        Create Trip $trip_p$.
        Assign $v$ and $p$ to $trip_p$ and $emt_p$.
        Chain $emt_p$ to Movements.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
    EndIf
End
```

Another approach, very similar to Algorithm 4, considers all buses during the assignment process, not just empty or incoming buses. This algorithm calculates the nearest vehicle by assessing the distance/time from non-empty buses to the waiting passenger, which includes the time required for the buses to reach their destinations, plus the time needed to transition from these destinations to the station where the waiting passenger is located (if the bus is incoming and its current destination is the waiting passenger's station, this transition time is counted as zero). For empty buses, the consideration is simply the current distance/time to the waiting passenger. By calculating and comparing the distance/time for all buses in the network to the waiting passenger, vehicles are an assigned accordingly. This method is an advanced refinement of Algorithm 4, theoretically enhancing both fairness and efficiency. However, the implementation of this algorithm presents certain challenges, particularly during the compilation phase. Given that the pseudocode for this algorithm closely resembles that of Algorithm 4, it is not detailed further here.

4.2 Dynamic Assignment

In the SNN assignment method, once a vehicle is assigned to a passenger, the assignment is immutable. This approach is somewhat unrealistic in real-world scenarios, primarily because the factors considered are often fewer than those actually impacting real-world situations, and the judgment of the "nearest vehicle" is based solely on distance or limited historical information about the road environment, which can be inaccurate.

More importantly, our algorithms are heuristic rather than exact optimization. As the time window progresses, although we provide the optimal solution at each individual time point—identifying and assigning the nearest vehicle—the solution is not optimal over a period. As buses continuously become available due to passengers alighting and new passengers appear at stations waiting for a ride, the overall optimal solution dynamically changes. To address this, our algorithm needs to incorporate the capability to dynamically adjust previous assignments.

One of the simplest forms of dynamic assignment involves considering already defined empty trips. Specifically, if a bus has an empty trip planned for picking up a passenger, and there happens to be another passenger or passengers needing a ride from the starting point to the destination of this empty trip, then the bus can accommodate these additional passengers. This approach is quite rational because, although it deviates from the FCFS principle, it does not adversely affect anyone's waiting time or interests. Implementing this strategy will undoubtedly enhance the efficiency of the APT system based on the existing algorithm framework. The specific pseudocode for this approach is outlined in Algorithm 5.

Algorithm 5 Add passengers to empty trips

```
Input: Graph $G = G(V, A)$;
        Predefined demands: number of passengers $n_p$, arriving time $t_p$,
            destination $s_p$;
            Vehicle set;
Output: All movements.
For every 20 simulation steps:
        SortStation ← sort by the longest waiting time of unassigned passengers.
        $s_i$ ← top (SortStation).
        $p$ ← longest waiting unassigned passenger at station $s$.
        If station $s_i$ has empty vehicle $v$ then:
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$.
                Chain $trip_p$ to Movements.
                Remove $p$ from $s_i$.unassigned_passengers.
        Else:
                Find nearest empty vehicle $v$. //the station where empty vehicle $v$ is
                                                    located is referred to as $s_j$
                Create EmptyTrip $emt_p$.
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$ and $emt_p$.
                Chain $emt_p$ to Movements.
                Chain $trip_p$ to Movements.
                Remove $p$ from $s_i$.unassigned_passengers.
                If there are passengers wishing to travel from $s_j$ to $s_i$:
                        Set P ← passengers wishing to travel from $s_j$ to $s_i$
                              //boarding based on waiting time, total number does not
                               exceed vehicle capacity
                        Remove passengers in set P from $s_j$.unassigned_passengers
                EndIf
        EndIf
End
```

A deeper reassignment goes beyond reassigning passengers and vehicles that merely have the same origin and destination; it involves reassigning all vehicles. Another approach is to dynamically find the optimal assignment in real-time. More specifically, the algorithm adjusts the assignment of passengers in already assigned vehicles based on updated real-time conditions, ensuring that the new assignment is better for each passenger than the original. This is equivalent to reassigning all passengers or vehicles during each assignment cycle, regardless of whether they were previously assigned, except for those who are already on board. This leads to the following code:

Algorithm 6 Reassignment

```
Input: Graph $G = G(V, A)$;
        Predefined demands: number of passengers $n_p$, arriving time $t_p$,
        destination $s_p$;
        Vehicle set;
Output: All movements.
For every 20 simulation steps:
    SortStation ← sort by the longest waiting time of unassigned passengers.
    While the number of unassigned passengers are not zero do:
        $s$ ← top (SortStation).
        $p$ ← longest waiting unassigned passenger at station $s$.
        If $p$.veh is empty:    //$p$ is not yet assigned to any vehicle before
            If station $s$ has empty vehicle $v$ then:
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$.
                Chain $trip_p$ to Movements.
                Remove $p$ from s.passengers.
            Else:
                Find nearest empty vehicle $v$.
                Create EmptyTrip $emt_p$.
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$ and $emt_p$.
                Chain $emt_p$ to Movements.
                Chain $trip_p$ to Movements.
                $p$.veh ← $v$
            EndIf
        Else:
            If station $s$ has empty vehicle $v$ then:
                Unassign the $emt_p$ and $trip_p$ that was originally
                assigned to $p$.veh.
                Create Trip $trip_p$.
```

```
                    Assign $v$ and $p$ to $trip_p$.
                    Chain $trip_p$ to Movements.
                    Remove $p$ from s.passengers.
                Else:
                    Find nearest empty vehicle $v$.
                    If $p$.veh$\neq v$:
                        Unassign the $emt_p$ and $trip_p$ that was
                        originally assigned to $p$.veh.
                        Create EmptyTrip $emt_p$.
                        Create Trip $trip_p$.
                        Assign $v$ and $p$ to $trip_p$ and $emt_p$.
                        Chain $emt_p$ to Movements.
                        Chain $trip_p$ to Movements.
                        $p$.veh $\leftarrow v$
                    EndIf
                EndIf
            EndIf
        EndWhile
    End
```

However, there is an unresolved issue with this algorithm. For empty vehicles already assigned to specific passengers and en route to pick them up, if a closer and faster vehicle becomes available, it is unclear how to handle the originally assigned empty vehicles. This algorithm follows a greedy principle, but it is likely to become suboptimal. Therefore, its actual performance needs to be observed during implementation.

4.3 Ride-Sharing Assignment

In APT system, ride-sharing is a crucial component. Although previous algorithms have incorporated dependent ride-sharing algorithms, where passengers can board buses that have vacancies and share the same origin and destination points, ride-sharing was not considered in the initial vehicle assignment. To enhance efficiency and assign limited bus resources to more passengers, it is essential to consider ride-sharing in the bus assignment process to stations. Specifically, we prefer to assign vehicles primarily to stations with a higher number of passengers sharing the same destination. The specific assignment method may vary depending on the factors considered.

A very simple approach is to assign passengers to vehicles by assigning all passengers with the same origin and destination to the same vehicle, instead of assigning only one specific passenger to a vehicle. This prevents subsequent vehicles from being assigned to passengers who share the same origin and destination, only to find that these passengers have already departed due to the ride-sharing mode, thus avoiding unnecessary empty trips and achieving a better assignment method. This method is clearly more efficient, as it incorporates the ride-sharing model during the assignment phase. Based on this idea, we wrote the pseudocode according to the previously mentioned Algorithm 2:

Algorithm 7

```
Input: Graph $G = G(V, A)$;
        Predefined demands: number of passengers $n_p$, arriving time $t_p$,
        destination $s_p$;
        Vehicle set;
Output: All movements.
For every 20 simulation steps:
        SortStation ← sort by the number of unassigned passengers.
        $s$ ← top (SortStation).
        $p$ ← longest waiting unassigned passenger at station $s$.
        If station $s$ has empty vehicle $v$ then:
                Create Trip $trip_p$.
                Assign $v$ and $p$ to $trip_p$.
                Chain $trip_p$ to Movements.
                Remove $p$ from s.unassigned_passengers.
                For all other unassigned passengers $p'$ with same origin and destination
                as $trip_p$:
                        assign $p'$ to $trip_p$
                        Remove $p'$ from s.unassigned_passengers.
        Else:
                Find nearest empty vehicle $v$.
                Create EmptyTrip $emt_p$.
```

```
            For all other unassigned passengers $p'$ with same origin and destination as $emt_p$:
                assign $p'$ to $emt_p$.
                Remove $p'$ from unassigned_passengers.
            Assign $v$ to $emt_p$.
            Create Trip $trip_p$.
            Assign $v$ and $p$ to $trip_p$
            Chain $emt_p$ to Movements.
            Chain $trip_p$ to Movements.
            Remove $p$ from s.unassigned_passengers.
            For all other unassigned passengers $p'$ with same origin and destination as $trip_p$:
                assign $p'$ to $trip_p$
                Remove $p'$ from s.unassigned_passengers.
        EndIf
End
```

Additionally, consideration of assigning passengers can be introduced when sorting stations. The widely used principles of sorting stations, such as FCFS or prioritizing stations with the maximum number of waiting passengers, tend to assign vehicles to stations with the longest waiting passengers or the most waiting passengers. However, an alternative approach could be to assign vehicles to stations with the highest number of passengers sharing the same origin and destination as one of the consideration factors. This could improve the overall efficiency of the transportation network. Algorithms 8-10 present the pseudocode for the SortStation function. Specifically, Algorithm 8 only considers the ride-sharing factor, prioritizing the assignment of vehicles to the groups with the most ride-sharing passengers.

Algorithm 8 SortStation Function1

```
Function: SortStation
For every station $s$:
    Group the passengers who are going to the same station.
    Count the number of passengers in each group.
    Split the group if the number of passengers in it is larger than vehicle capacity.
    Define group set as $g = \{1, \ldots, k\}$, number of passengers in each group as
```

```
        $N = \{n_1, \dots, n_k\}$.
        s.Index ← largest group number.
    End
    Sort station according to s.Index.
    EndFunction
```

In Algorithm 9, we not only consider the groups with the most ride-sharing passengers, but also the total number of waiting passengers at a station. To simultaneously account for both factors, we use a specific index, $Index = \sum_{i=1}^{k} e^{n_i}$, in this pseudocode, where e denotes the base of the natural logarithm.

Here is an example to explain this index in detail: If Station 1 has four passengers, with two of them wanting to go to the same destination and the other two going to different destinations, the index is calculated as $Index = e^2 + e + e$. On the other hand, if Station 2 has six passengers, each going to different destinations, the index is calculated as $Index = 6e$. Since $e^2 + 2e < 6e$, vehicles will be prioritized for Station 2.

This shows that the order of vehicle assignment to stations is determined by both factors. In the formula we use, the ride-sharing factor has a larger weight because it has an exponential impact, while the total number of waiting passengers has a relatively smaller influence. Of course, the mathematical expression can be modified to other forms, such as a weighted sum of the two factors.

Algorithm 9 SortStation Function2

```
Function: SortStation
For every station $s$:
    Group the passengers who are going to the same station.
    Count the number of passengers in each group.
    Split the group if the number of passengers in it is larger than vehicle capacity.
    Define group set as $g = \{1, \dots, k\}$, number of passengers in each group as
    $N = \{n_1, \dots, n_k\}$.
    s.Index ← $\sum_{i=1}^{k} e^{n_i}$
End
Sort station according to s.Index.
EndFunction
```

Similarly, in Algorithm 10, we consider both the groups with the most ride-sharing passengers and the longest waiting time as factors. The mathematical expression we use is: $Index = \sum_{i=1}^{k} {n_i}^{w_i}$. However, the exact mathematical expression to be used should be determined based on actual simulations; this is just a potential example.

Algorithm 10 SortStation Function3

```
Function: SortStation
For every station $s$:
        Group the passengers who are going to the same station.
        Count the number of passengers in each group.
        Split the group if the number of passengers in it is larger than vehicle capacity.
        Define group set as $g = \{1, \dots, k\}$, number of passengers in each group as $N = \{n_1, \dots, n_k\}$.
        Calculate the waiting time of the longest waiting passenger in each group, define it as $W = \{w_1, \dots, w_k\}$.
        s.Index ← $\sum_{i=1}^{k} {n_i}^{w_i}$.
End
Sort station according to s.Index.
EndFunction
```

4.4 Algorithms to be tested

There are many algorithms mentioned above, and it is not feasible to test each one in this report. Therefore, this report will test the most representative algorithms.

First, this paper will test the basic algorithms, which are Algorithm 1(MaxWaitSimple) and Algorithm 2 (MaxNrPassengers) as mentioned earlier.

Subsequently, we will test Algorithm MaxWaitSimple-B and MaxNrPassengers-B. Algorithm MaxWaitSimple-B integrates aspects from Algorithms 1, 5, and 7. Specifically, it prioritizes the number of unassigned passengers when sorting stations, allocating vehicles to stations with the highest number of unassigned passengers (Algorithm 1). Additionally, if there are passengers with the same origin and destination, they are added to empty trips (Algorithm 5). Furthermore, during the assignment phase, it considers ride-sharing by assigning all passengers with the same origin and destination to the same vehicle, instead of assigning only one specific passenger to a vehicle (Algorithm 7). This is a comprehensive algorithm that considers multiple factors, including dynamic assignment and ride-sharing. Similarly, for MaxNrPassengers-B, we integrates aspects from Algorithms 2, 5, and 7.

Below, the pseudocode for algorithm MaxWaitSimple-B will be displayed.

Algorithm MaxWaitSimple-B

```
Input: Graph $G = G(V, A)$;
          Predefined demands: number of passengers $n_p$, arriving time $t_p$;
          Vehicle set;
Output: All movements.
For every 20 simulation steps:
          SortStation ← sort by the number of unassigned passengers.
          $s$ ← top (SortStation).
          $p$ ← longest waiting unassigned passenger at station $s$.
          If station $s$ has empty vehicle $v$ then:
                    Create Trip $trip_p$.
                    Assign $v$ and $p$ to $trip_p$.
                    Chain $trip_p$ to Movements.
                    Remove $p$ from s.unassigned_passengers.
                    For all other unassigned passengers $p'$ with same origin and destination
                    as $trip_p$:
                              assign $p'$ to $trip_p$
                              Remove $p'$ from s.unassigned_passengers.
          Else:
                    Find nearest empty vehicle $v$.
                    Create EmptyTrip $emt_p$.
                    For all other unassigned passengers $p'$ with same origin and destination
                    as $emt_p$:
                              assign $p'$ to $emt_p$.
                              Remove $p'$ from unassigned_passengers.
                    Assign $v$ to $emt_p$.
                    Create Trip $trip_p$.
```

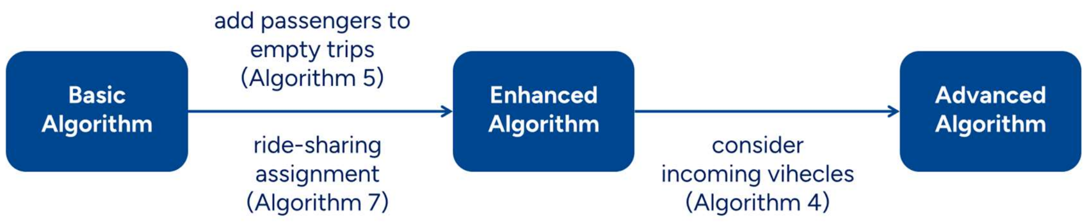
```
        Assign $v$ and $p$ to $trip_p$
        Chain $emt_p$ to Movements.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
        For all other unassigned passengers $p'$ with same origin and destination
        as $trip_p$:
            assign $p'$ to $trip_p$
            Remove $p'$ from s.unassigned_passengers.
    EndIf
End
```

Algorithm MaxWaitSimple-C and Algorithm MaxNrPassengers-C are advanced versions of Algorithm MaxWaitSimple-B and Algorithm MaxNrPassengers-B, respectively. These algorithms integrate the methodologies of Algorithm 4 from the previous chapter. When assigning vehicles, they consider not only empty vehicles but also incoming vehicles. Specifically, if passengers are waiting at station s_i, the algorithm first checks for any empty vehicles at station s_i. If none are available, it then considers whether any unassigned vehicles carrying passengers are en route to station s_i as their destination. If neither condition is met, the nearest empty vehicle is considered.

Algorithm MaxWaitSimple-C

```
Input: Graph $G = G(V, A)$;
    Predefined demands: number of passengers $n_p$, arriving time $t_p$;
    Vehicle set;
Output: All movements.
For every 20 simulation steps:
    SortStation ← sort by the number of unassigned passengers.
    $s$ ← top (SortStation).
    $p$ ← longest waiting unassigned passenger at station $s$.
    If station $s$ has empty vehicle $v$ then:
        Create Trip $trip_p$.
        Assign $v$ and $p$ to $trip_p$.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
        For all other unassigned passengers $p'$ with same origin and destination
        as $trip_p$:
            assign $p'$ to $trip_p$
            Remove $p'$ from s.unassigned_passengers
    Else If station $s$ has incoming vehicle $v$' that will be empty after
        Create Trip $trip_p$.
        Assign $v'$ and $p$ to $trip_p$.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
        For all other unassigned passengers $p'$ with same origin and destination
        as $trip_p$:
            assign $p'$ to $trip_p$
            Remove $p'$ from s.unassigned_passengers
    Else
        Find nearest empty vehicle $v$.
        Create EmptyTrip $emt_p$.
        For all other unassigned passengers $p'$ with same origin and destination
        as $emt_p$.:
            assign $p'$ to $emt_p$.
            Remove $p'$ from unassigned_passengers.
        Assign $v$ to $emt_p$.
        Create Trip $trip_p$.
        Assign $v$ and $p$ to $trip_p$
        Chain $emt_p$ to Movements.
        Chain $trip_p$ to Movements.
        Remove $p$ from s.unassigned_passengers.
        For all other unassigned passengers $p'$ with same origin and destination
        as $trip_p$:
            assign $p'$ to $trip_p$
            Remove $p'$ from s.unassigned_passengers.
    EndIf
End
```

Figure 8 demonstrates the logic when deciding which algorithm and which feature to be tested. We first test the two basic algorithm, then we add two features to the basic algorithm and form two enhanced algorithm. We add one more feature to the enhanced algorithm and gets two advanced algorithms.



Basic Algorithm
add passengers to empty trips (Algorithm 5)
ride-sharing assignment (Algorithm 7)
Enhanced Algorithm
consider incoming vihecles (Algorithm 4)
Advanced Algorithm

Figure 8. Algorithm to be tested.

5 Numerical Experiment

In this study, we selected some of the previously mentioned algorithms and tested them using the VIPSim simulator on the four networks discussed in Chapter 3.

Our simulation utilizes several key performance indicators (KPIs) to evaluate the efficiency of each algorithm across different network environments. These KPIs include: the number of passengers served, maximum passenger wait time, average passenger wait time, average passenger travel distance, total passenger travel distance, empty vehicle distance ratio, average vehicle occupancy, and average queue length. Detailed results for these KPIs will be displayed in the tables within the Appendix. In the main body of the report, we primarily focus on analyzing two crucial metrics: the maximum passenger waiting time and the average passenger waiting time.

5.1 Basic Algorithm

First, we tested Algorithm 1 and Algorithm 2. This was to explore whether it is better to assign vehicles to the longest waiting passenger (following the FCFS principle) or to assign them to the station with the highest number of unassigned waiting passengers. A preliminary prediction is that allocating to the longest waiting passenger would be fairer and minimize the maximum wait time, while prioritizing the station with the most unassigned waiting passengers would increase efficiency. To test our hypothesis, we obtained the results shown in Figures 9 and 10, where "simplesquare" represents the symmetric simple network, while "simplesquare(2)" represents the asymmetric simple network. Figure 9 shows the comparison results of the two algorithms in terms of the maximum passenger wait time, while Figure 10 presents the comparison results in terms of the average passenger wait time.

Our initial prediction was that Algorithm 1 (MaxWaitSimple), which assigns vehicles to the longest waiting passenger, would result in a shorter maximum passenger wait time. However, Figure 9 shows the opposite. Except for the "simplesquare" network, where the symmetry likely minimizes the impact of different assignment strategies, we found that Algorithm 2 (MaxNrPassengers), which prioritizes stations with the highest number of waiting passengers, outperformed Algorithm 1 in both maximum passenger wait time and average passenger wait time.

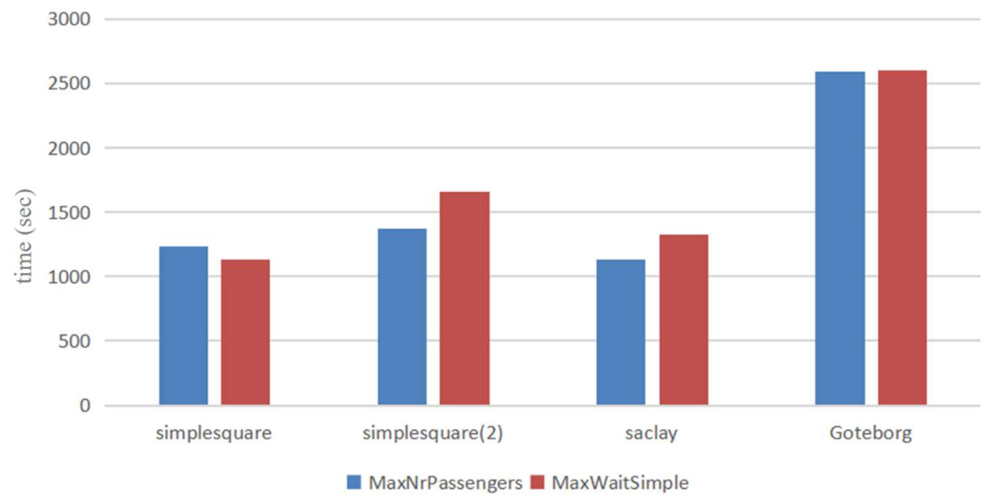


Figure 9. Comparison of Algorithm MaxNrPassengers and Algorithm MaxWaitSimple under the metric of maximum passenger waiting time.

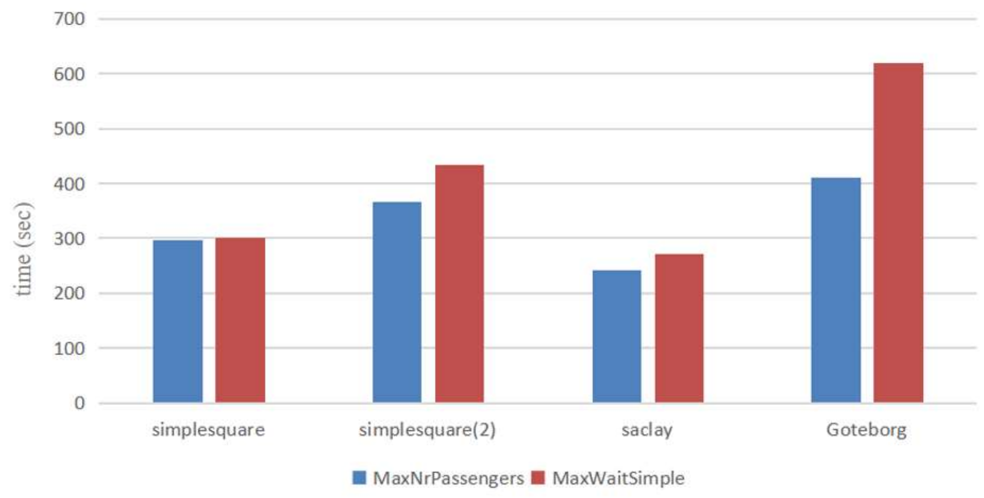


Figure 10. Comparison of Algorithm MaxNrPassengers and Algorithm MaxWaitSimple under the metric of average passenger waiting time.

A detailed analysis of vehicle assignments and movements suggests that the reason for this unexpected result is a decline in overall efficiency (evidenced by an increase in average passenger wait time), which negatively impacts all metrics, including maximum passenger wait time. In other words, although Algorithm 1 aimed to be fairer by addressing the longest waiting passenger, the decrease in overall efficiency led to worse outcomes for everyone.

Therefore, from this comparison, we conclude that in our very basic assignment algorithms, prioritizing the number of unassigned passengers at a station is a more efficient method. Additionally, when emphasizing fairness, the overall system efficiency may suffer. More importantly, we found that a decline in overall system efficiency can lead to worse outcomes for everyone, including those who were supposed to be prioritized and favored.

5.2 Enhanced Algorithm

Subsequently, we executed Algorithm MaxNrPassengers-B.

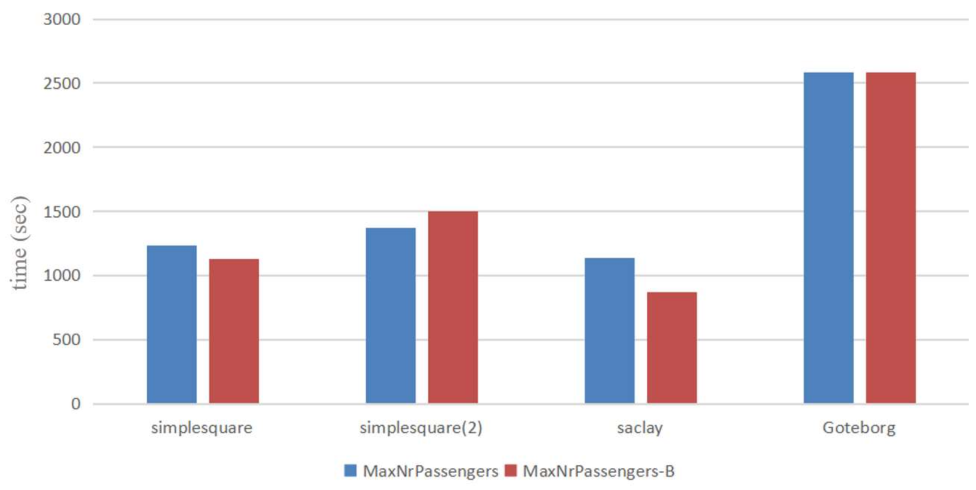


Figure 11. Comparison of Algorithm MaxNrPassengers and Algorithm MaxNrPassengers-B under the metric of maximum passenger waiting time.

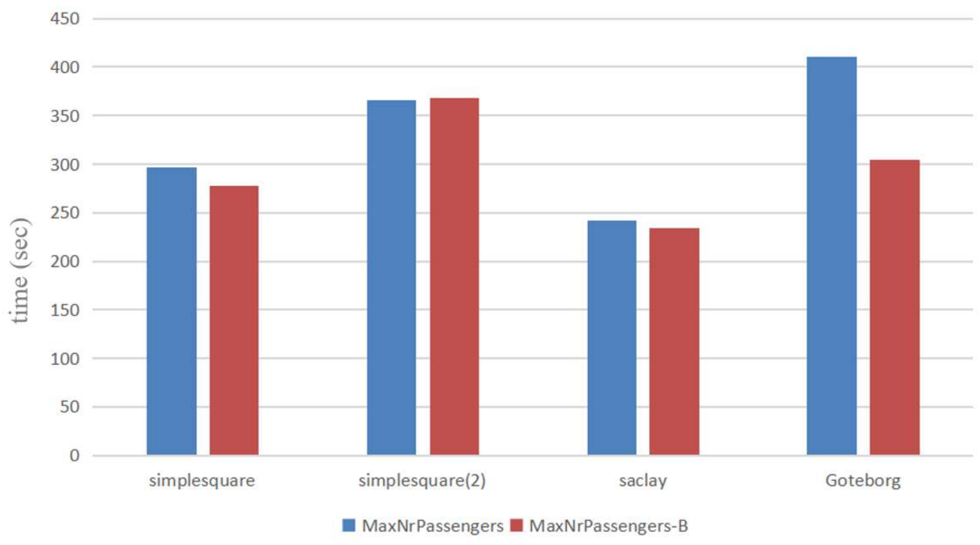


Figure 12. Comparison of Algorithm MaxNrPassengers and Algorithm MaxNrPassengers-B under the metric of average passenger waiting time.

The test results are shown in Figures 11 and 12. We observed that, apart from the "SimpleSquare(2)" network where Algorithm MaxNrPassengers-B performs marginally worse than Algorithm 2 (MaxNrPassengers), Algorithm MaxNrPassengers-B consistently outperforms Algorithm 2 (MaxNrPassengers) in all other network configurations. This finding is in line with our initial expectations, which posited that Algorithm MaxNrPassengers-B would yield better results in practical network scenarios.

Similarly, we tested Algorithm MaxWaitSimple-B, which integrates aspects from Algorithms 1, 5, and 7. Similar to Algorithm MaxNrPassengers-B. When comparing the results of Algorithm MaxWaitSimple-B to those of Algorithm 1 (MaxWaitSimple), we observed different outcomes, as shown in Figures 13 and 14. The results distinctly show that Algorithm MaxWaitSimple-B outperforms Algorithm 1 (MaxWaitSimple) across all four networks, with the improvements being particularly significant in the "Goteborg" network. Moreover, the advancements of Algorithm MaxWaitSimple-B over Algorithm 1 (MaxWaitSimple) are more substantial than those of Algorithm MaxNrPassengers-B over Algorithm 2 (MaxNrPassengers). This marked improvement is likely due to Algorithm 1 (MaxWaitSimple) solely adhering to a FCFS basis, thus neglecting the overall efficiency of the network. The new features integrated into Algorithm MaxWaitSimple-B have significantly rectified this oversight, enhancing overall network performance.

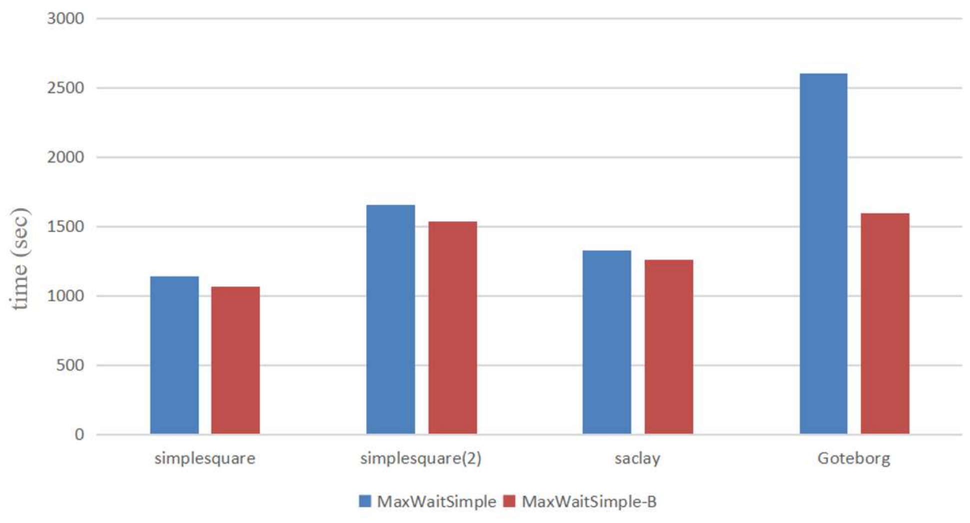


Figure 13. Comparison of Algorithm MaxWaitSimple and Algorithm MaxWaitSimple-B under the metric of maximum passenger waiting time.

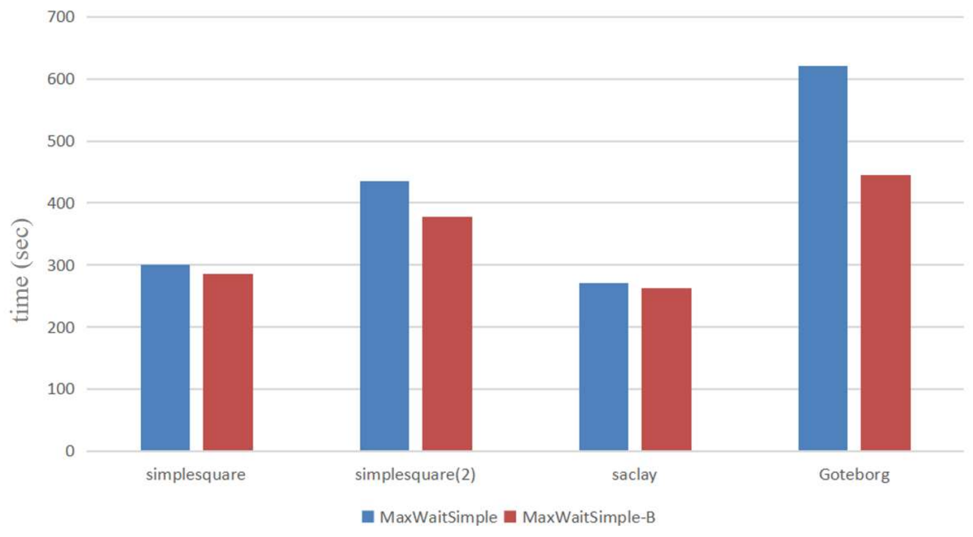


Figure 14. Comparison of Algorithm MaxWaitSimple and Algorithm MaxWaitSimple-B under the metric of average passenger waiting time.

Then, compare Algorithm MaxNrPassengers-B with Algorithm MaxWaitSimple-B. Unlike Algorithm 2 (MaxNrPassengers), which surpasses Algorithm 1 (MaxWaitSimple) in both maximum and average passenger waiting times, Algorithm MaxNrPassengers-B performs better than Algorithm MaxWaitSimple-B in terms of average passenger waiting time, yet it underperforms in maximum passenger waiting time.

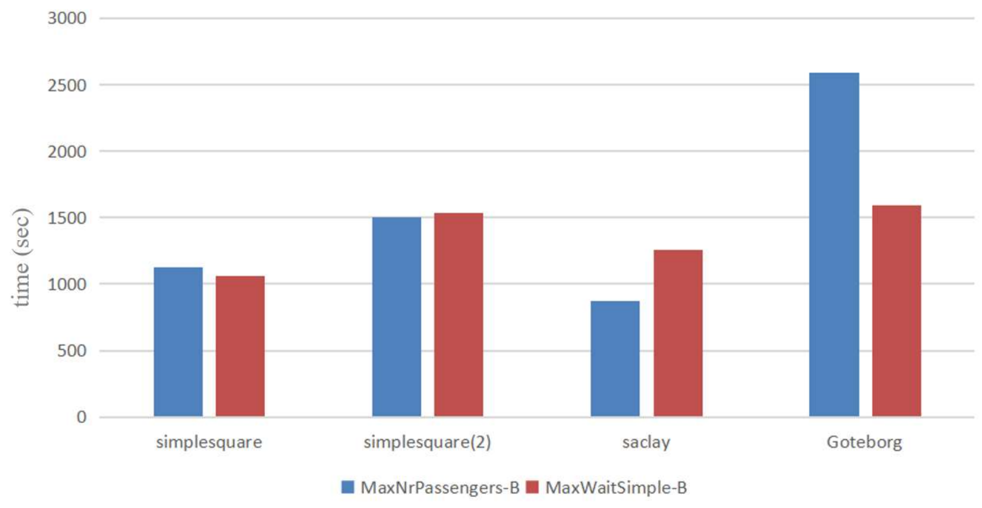


Figure 15. Comparison of Algorithm MaxNrPassengers-B and Algorithm MaxWaitSimple-B under the metric of maximum passenger waiting time.

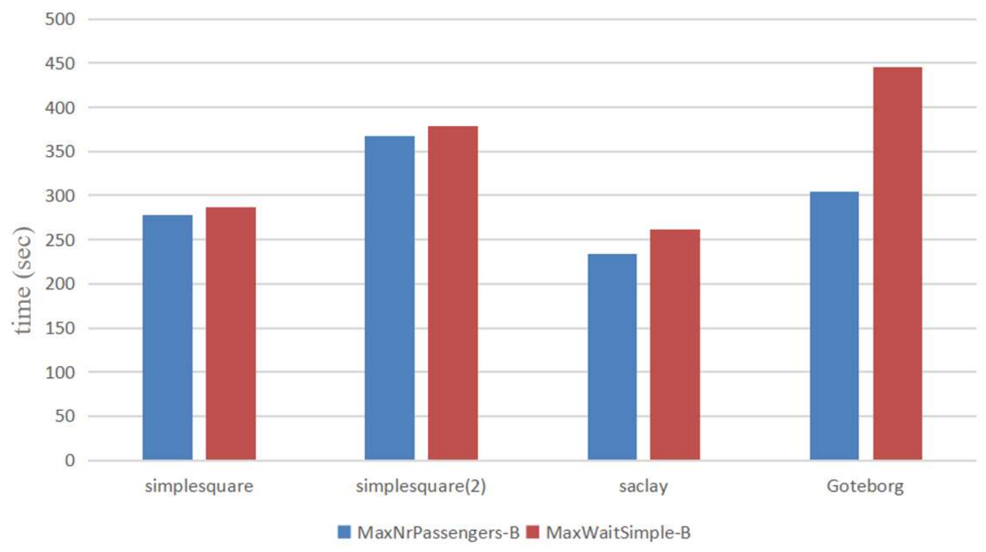


Figure 16. Comparison of Algorithm MaxNrPassengers-B and Algorithm MaxWaitSimple-C under the metric of average passenger waiting time.

This aligns with our initial expectations regarding the two assignment strategies. Assigning vehicles to the longest waiting passenger, as employed by Algorithm MaxWaitSimple-B, reduces the maximum passenger waiting time, thereby achieving a fairer distribution of wait times. Conversely, assigning vehicles to stations with the highest number of unassigned waiting passengers, a strategy used by Algorithm MaxNrPassengers-B, tends to improve the overall operational efficiency of the network. This analysis supports the theoretical basis for the designed algorithms and their intended impacts on network performance.

5.3 Advanced Algorithm

The subsequent section introduces the test results for Algorithm MaxNrPassengers-C and Algorithm MaxWaitSimple-C. These algorithms build on the foundation of Algorithm MaxNrPassengers-B and Algorithm MaxWaitSimple-B by adding the feature "add passengers to empty trips", which was previously mentioned in Algorithm 5. Intuitively, this approach aims to reduce the mileage of empty vehicles and enhance the overall efficiency of the network. However, it may increase the waiting time for some passengers, as the nearest empty vehicle might arrive faster, but to avoid dispatching empty vehicles, an incoming vehicle is assigned instead.

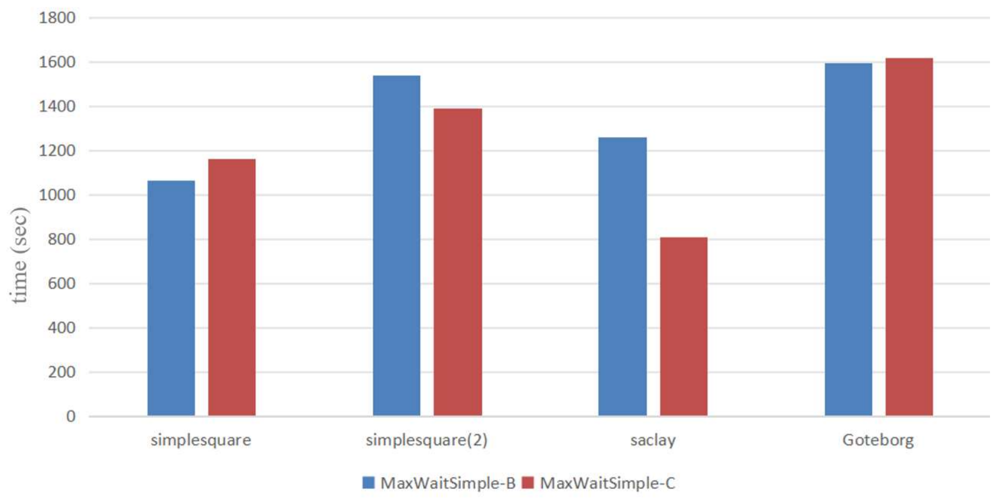


Figure 17. Comparison of Algorithm MaxWaitSimple-B and Algorithm MaxWaitSimple-C under the metric of maximum passenger waiting time.

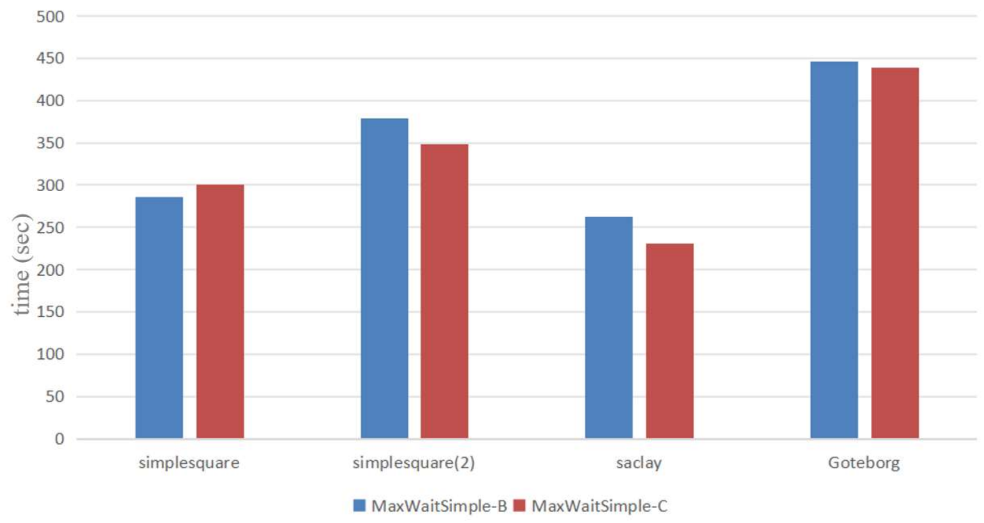


Figure 18. Comparison of Algorithm MaxWaitSimple-B and Algorithm MaxWaitSimple-C under the metric of average passenger waiting time.

To validate our hypotheses, numerical experiments were conducted to compare Algorithm MaxWaitSimple-B and Algorithm MaxWaitSimple-C. The results, illustrated in Figures 17 and 18, revealed unexpected findings. Contrary to our initial assumptions, it was not possible to determine which algorithm performed better under the metrics of maximum passenger waiting time and average passenger waiting time. Each algorithm excelled in different networks. Additionally, we observed that an algorithm could perform either better or worse on both the maximum and average passenger waiting times. This characteristic was consistent with the results for the basic algorithm discussed in Section 5.1, and the underlying reasons have been discussed previously and will not be reiterated here.

5.4 Overall Test Results

Finally, a comparative analysis of all the algorithms tested in this study is presented in Figures 19 and 10. Algorithm MaxWaitSimple demonstrated the poorest performance, ranking lowest in both efficiency (measured by average passenger waiting time) and maximum passenger waiting time. In contrast, Algorithm MaxNrPassengers-B was identified as the most efficient in optimizing network efficiency. Determining the best performer in terms of maximum passenger waiting time proved to be challenging, as no clear leader emerged from the analysis.

Furthermore, our findings indicate that the impact of additional features varies depending on the base algorithm. For instance, incorporating the "incoming vehicles" feature significantly improved the performance of Algorithm MaxWaitSimple-B compared to its effect on Algorithm MaxNrPassengers-B. Therefore, it is essential not to assume intuitively that any new feature will enhance efficiency. Instead, improvements should be assessed based on testing outcomes across multiple networks to determine the specific enhancements to a particular algorithm.

Balancing efficiency and fairness is crucial. The overall network's FCFS allocation should be based on the metric of maximum passenger waiting time, which we believe best satisfies the principle of fairness. However, results reveal that, among algorithms with the same functionalities, those allocating vehicles based on the maximum number of passengers often exhibit higher efficiency than those based on maximum passenger waiting time. This necessitates an understanding that efficiency may come at the cost of some fairness, requiring a trade-off between the two.

Simultaneously, our assumption is that there should be a trade-off between maximum passenger waiting time and average passenger waiting time. When we focus more on reducing maximum passenger waiting time, the average passenger waiting time tends to increase. However, in practice, the trends of maximum passenger waiting time and average passenger waiting time are generally similar. That is to say, when the average passenger waiting time is low, the maximum passenger waiting time is also typically reduced. We believe the underlying logic is that when the overall efficiency of the network (average passenger waiting time) improves, the benefits extend to most passengers, reducing the waiting time for the majority and thereby decreasing the maximum waiting time as well. Thus, we consider that enhancing overall efficiency is a more crucial factor in the development of transportation network algorithms.

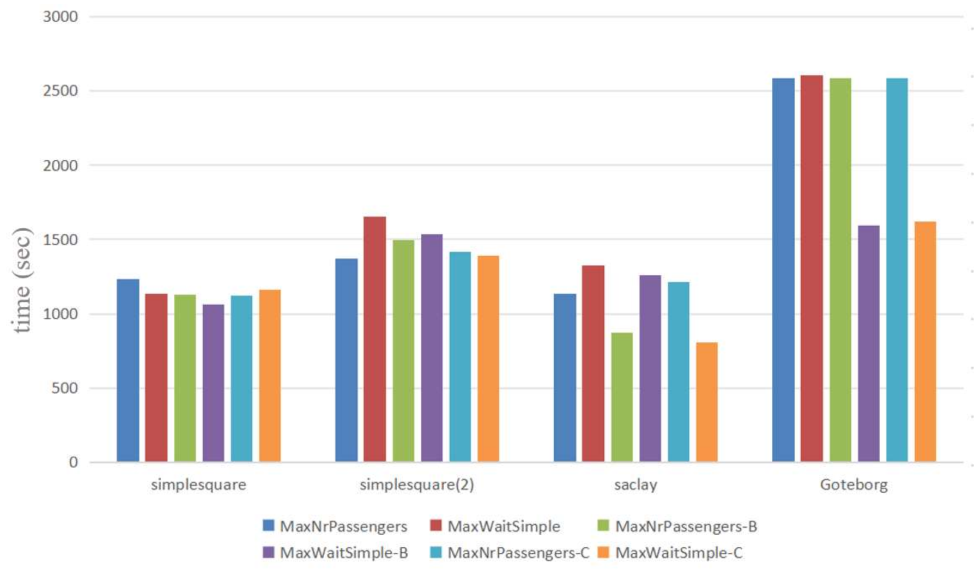


Figure 19. Comparison of all algorithms under the metric of maximum passenger waiting time.

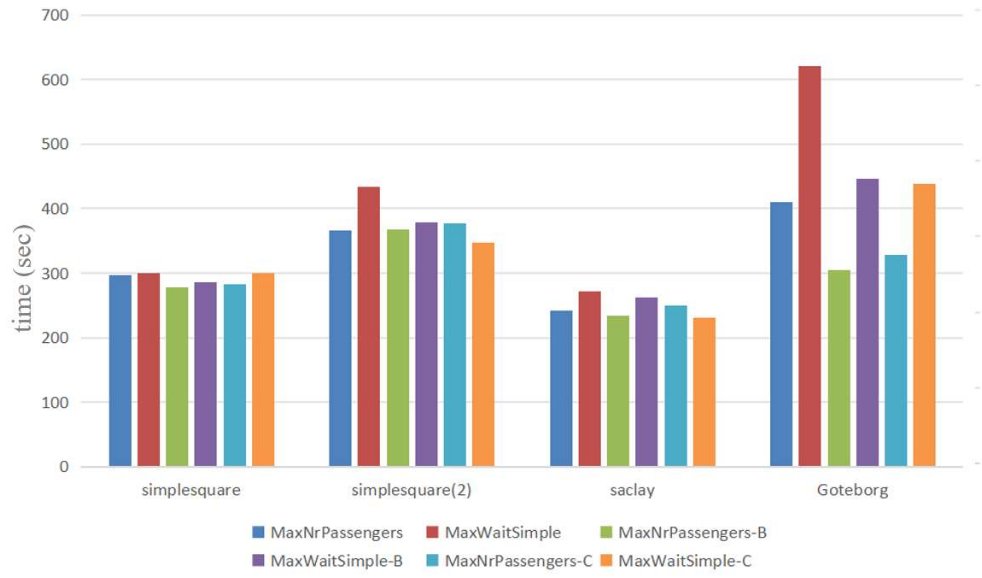


Figure 20. Comparison of all algorithms under the metric of average passenger waiting time.

Evaluating the network from the passenger perspective using metrics such as maximum passenger waiting time and average passenger waiting time is very effective, as they distinctly reflect the passenger experience. From the operator perspective, metrics like the Ratio of empty vehicle kilometers and the average number of passengers per loaded vehicle are also worth discussing and are often important efficiency metrics as well.

Figure 21 shows a comparison of six algorithms under the metric of Ratio of empty vehicle kilometers, where we find that in complex networks, the algorithm MaxNrPassengers-B is optimal. This aligns with the conclusions under the metric of average passenger waiting time. However, in simpler networks, the algorithm MaxNrPassengers-C performs slightly better than MaxNrPassengers-B.

Figure 22 presents the average number of passengers per loaded vehicle. We find that the differences between algorithms on this metric are not significant. Moreover, the average number of passengers per loaded vehicle is far less than the vehicle capacity of 10. This also indicates that our algorithms need further improvement in practical ride-sharing scenarios (not just considering ride-sharing factors at the assignment stage). Our algorithms and functionalities have not yet addressed this aspect, which could be incorporated into future research. For instance, vehicles with available seats could pick up passengers en route or plan routes that slightly detour to accommodate more passengers, thus increasing ride-sharing opportunities. These aspects could be considered for further development.

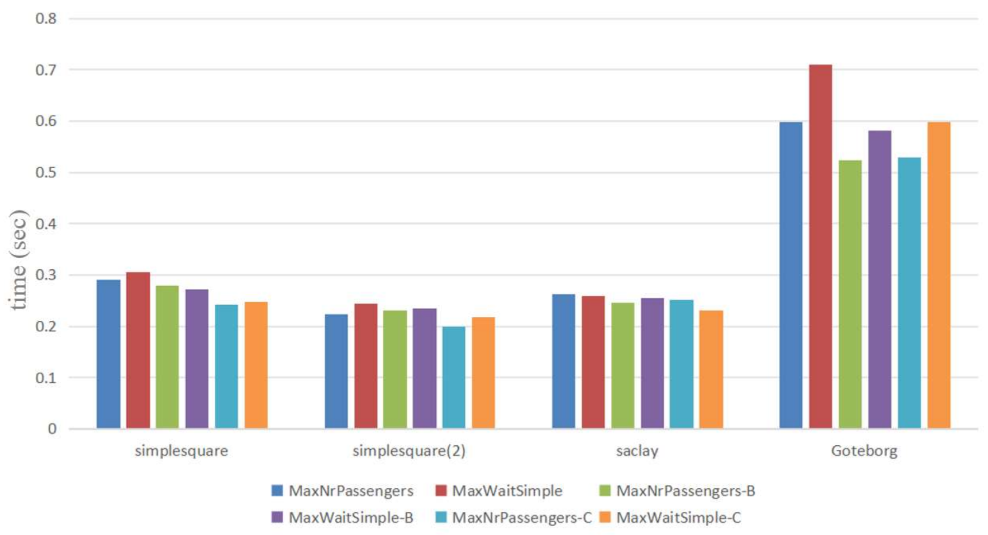


Figure 21. Comparison of all algorithms under the metric of ratio of empty vehicle km.

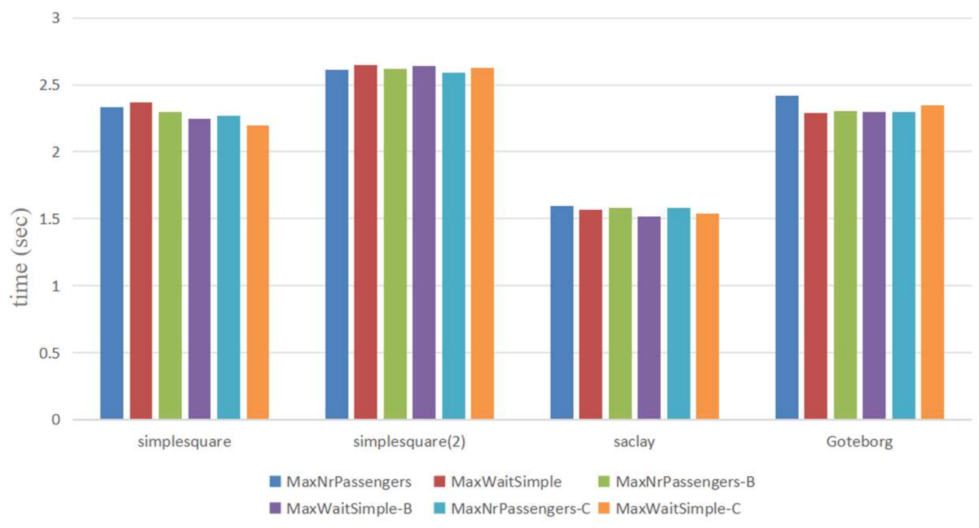


Figure 22. Comparison of all algorithms under the metric of average number of passengers per loaded vehicle.

6 Conclusion

6.1 Summary

This report aims to introduce the concept of APT into urban transit systems to enhance flexibility, cost-efficiency, and overall performance. The successful implementation of APT requires sophisticated vehicle assignment algorithms that distribute vehicles to passengers based on specific rules. Therefore, this study summarizes and proposes several effective assignment algorithms and evaluates their performance through simulation experiments conducted on a traffic simulator.

Specifically, the report presents several innovative approaches to algorithm improvement and ultimately tests six algorithms. The test scenarios are carried out on four traffic networks within the VIPSim simulator, two of which are basic networks, while the other two are more complex, real-world networks. The findings identify the most efficient algorithm for network operation and offer several insightful conclusions. For example, (1) in basic algorithms, using average passenger waiting time as a metric for assignment achieves higher network efficiency than using maximum passenger waiting time; (2) the effectiveness of different algorithms varies across networks with different characteristics; (3) the impact of new features differs depending on the base algorithm, suggesting that features should be tailored specifically for each algorithm. These guidelines provide valuable recommendations for the future implementation of APT.

6.2 Future Direction

Additionally, this report identifies several promising yet unexplored research areas:

（1） Due to time constraints, not all algorithms proposed in Chapter 4 were tested. Future studies should aim to test these algorithms using simulators to better understand their effectiveness.

（2） The algorithms tested in this study incorporate multiple features, such as ride-sharing, adding passengers to empty trips, and considering incoming vehicles.

However, the specific impact of each feature on algorithm performance is unknown. Future research should test these features individually and in various combinations to ascertain their effects on network operations.

（3） Most allocation algorithms considered in this report are heuristic. Future research could develop exact optimization algorithms for simple networks to compare with our heuristic algorithms, providing a clearer perspective on their relative strengths and weaknesses.

（4） The current allocation algorithms do not address the routing of passengers after boarding, typically transporting groups with the same origin and destination without considering en-route pickups. Practically, vehicles with available capacity could pick up passengers at intermediate stops or slightly detour to maximize ride-sharing. This aspect could be explored in future expansions.

（5） This study was conducted on only four networks, which is insufficient to analyze the relationship between algorithms and network characteristics thoroughly. Future studies should expand the number of test networks, selecting those with distinctive features to evaluate different algorithms comprehensively.

（6） Demand significantly influences algorithm performance; varying demand characteristics may necessitate different algorithms. This area warrants further investigation.

6.3 Sustainability

Sustainability is a crucial consideration in contemporary development. The United Nations has established 17 Sustainable Development Goals (SDGs), a global action plan aimed at achieving peace and prosperity while protecting the planet's natural environment by 2030.

This report contributes to sustainability through its focus on the APT system, which aligns perfectly with "Goal 11: Sustainable Cities and Communities." The APT system represents a more advanced, intelligent, and user-centric public transportation system that can enhance urban mobility services. And it has the potential to provide efficient first- and last-mile connections, thereby serving a broader demographic, including vulnerable groups, and fully meeting the requirements of sustainable development.

Moreover, this report also contributes to "Goal 13: Climate Action." It aims to improve the efficiency of the APT system by leveraging smart assignment algorithms to increase ride-sharing and reduce empty vehicle trips. The proposed enhanced and advanced algorithms help achieve this by minimizing unnecessary travel, reducing fuel consumption and emissions while maintaining or improving service quality. This makes the APT system more environmentally friendly and supportive of sustainability initiatives.

Reference

Agatz, N.A.H. *et al.* (2011) ‘Dynamic ride-sharing: A simulation study in metro Atlanta’, *Transportation Research Part B: Methodological*, 45 `(9). Available at: https://doi.org/10.1016/j.trb.2011.05.017.

Alonso-Mora, J. *et al.* (2017) ‘On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment’, *Proceedings of the National Academy of Sciences of the United States of America*, 114(3). Available at: https://doi.org/10.1073/pnas.1611675114.

Alsabaan, M., Naik, K. and Khalifa, T. (2013) ‘Optimization of fuel cost and emissions using V2V communications’, *IEEE Transactions on Intelligent Transportation Systems*, 14(3). Available at: https://doi.org/10.1109/TITS.2013.2262175.

Arslan, G., Marden, J.R. and Shamma, J.S. (2007) ‘Autonomous vehicle-target assignment: A game-theoretical formulation’, *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 129(5). Available at: https://doi.org/10.1115/1.2766722.

Carrese, F. *et al.* (2023) ‘The Integration of Shared Autonomous Vehicles in Public Transportation Services: A Systematic Review’, *Sustainability (Switzerland)*. Available at: https://doi.org/10.3390/su151713023.

Chen, T.D., Kockelman, K.M. and Hanna, J.P. (2016) ‘Operations of a shared, autonomous, electric vehicle fleet: Implications of vehicle & charging infrastructure decisions’, *Transportation Research Part A: Policy and Practice*, 94. Available at: https://doi.org/10.1016/j.tra.2016.08.020.

Clavijo, M., Jiménez, F. and Naranjo, J.E. (2023) ‘The Development and Prospects of Autonomous Driving Technology’, *Applied Sciences (Switzerland)*. Available at: https://doi.org/10.3390/app13095377.

Cokyasar, T. and Larson, J. (2020) ‘Optimal assignment for the single-household shared autonomous vehicle problem’, *Transportation Research Part B: Methodological*, 141. Available at: https://doi.org/10.1016/j.trb.2020.09.003.

Furda, A. *et al.* (2010) ‘Improving safety for driverless city vehicles: Real-time communication and decision making’, in *IEEE Vehicular Technology Conference*. Available at: https://doi.org/10.1109/VETECS.2010.5494179.

Hyland, M. and Mahmassani, H.S. (2018) ‘Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests’, *Transportation Research Part C: Emerging Technologies*, 92. Available at: https://doi.org/10.1016/j.trc.2018.05.003.

Iclodean, C., Cordos, N. and Varga, B.O. (2020) ‘Autonomous shuttle bus for public transportation: A review', *Energies*. Available at: https://doi.org/10.3390/en13112917.

Jordan, W.C., Llc, J.A. and Scarborough, B.A. (2013) 'TRANSFORMING PERSONAL MOBILITY Lawrence D. Burns, Director, Program on Sustainable Mobility', *Broadway* [Preprint].

Kyriakidis, M., Happee, R. and De Winter, J.C.F. (2015) 'Public opinion on automated driving: Results of an international questionnaire among 5000 respondents', *Transportation Research Part F: Traffic Psychology and Behaviour*, 32. Available at: https://doi.org/10.1016/j.trf.2015.04.014.

Lam, A.Y.S., Leung, Y.W. and Chu, X. (2016) 'Autonomous-Vehicle Public Transportation System: Scheduling and Admission Control', *IEEE Transactions on Intelligent Transportation Systems*, 17(5). Available at: https://doi.org/10.1109/TITS.2015.2513071.

Levin, M.W. *et al.* (2017) 'A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application', *Computers, Environment and Urban Systems*, 64. Available at: https://doi.org/10.1016/j.compenvurbsys.2017.04.006.

Litman, T. (2014) 'Autonomous Vehicle Implementation Predictions: Implications for Transport Planning', *Transportation Research Board Annual Meeting*, 42(2014). Available at: https://doi.org/10.1613/jair.301.

Pakusch, C. and Bossauer, P. (2017) 'User acceptance of fully autonomous public transport', in *ICETE 2017 - Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*. Available at: https://doi.org/10.5220/0006472900520060.

Poinsignon, F. *et al.* (2022) 'Autonomous vehicle fleets for public transport: scenarios and comparisons', *Green Energy and Intelligent Transportation*, 1(3). Available at: https://doi.org/10.1016/j.geits.2022.100019.

Santi, P. *et al.* (2014) 'Quantifying the benefits of vehicle pooling with shareability networks', *Proceedings of the National Academy of Sciences of the United States of America*, 111(37). Available at: https://doi.org/10.1073/pnas.1403657111.

Simonetto, A., Monteil, J. and Gambella, C. (2019) 'Real-time city-scale ridesharing via linear assignment problems', *Transportation Research Part C: Emerging Technologies*, 101. Available at: https://doi.org/10.1016/j.trc.2019.01.019.

Soteropoulos, A., Berger, M. and Ciari, F. (2019) 'Impacts of automated vehicles on travel behaviour and land use: an international review of modelling studies', *Transport Reviews*, 39(1). Available at: https://doi.org/10.1080/01441647.2018.1523253.

Tafidis, P. *et al.* (2022) 'Safety implications of higher levels of automated vehicles: a scoping review', *Transport Reviews*, 42(2). Available at:

https://doi.org/10.1080/01441647.2021.1971794.

Tafreshian, A. and Masoud, N. (2020) ‘Trip-based graph partitioning in dynamic ridesharing’, *Transportation Research Part C: Emerging Technologies*, 114. Available at: https://doi.org/10.1016/j.trc.2020.02.008.

Xing, Y. *et al.* (2021) ‘Toward human-vehicle collaboration: Review and perspectives on human-centered collaborative automated driving’, *Transportation Research Part C: Emerging Technologies*, 128. Available at: https://doi.org/10.1016/j.trc.2021.103199.

Zhang, W. *et al.* (2015) ‘Exploring the impact of shared autonomous vehicles on urban parking demand: An agent-based simulation approach’, *Sustainable Cities and Society*, 19. Available at: https://doi.org/10.1016/j.scs.2015.07.006.

Appendix

Table 1. KPIs of algorithm MaxNrPassengers and algorithm MaxWaitSimple on "simplesquare" network.

KPI	MaxNrPassengers	MaxWaitSimple
passengersServed	297	299
passengersNotServed	2	0
maxPassengerWait	1230.417693	1136.2
averagePassengerWait	297.1695203	300.2862202
averagePassengerTripTime	124.9324324	127.547973
averagePassengerKm	0.974917953	0.974986697
totalPassengerKm	288.5757141	288.5960624
vehicleKmLoaded	123.7837909	121.8244343
vehicleKmEmpty	50.72518633	53.64430304
vehicleKmEmptyRatio	0.290673793	0.305720004
passengersPerLoadedVehicle	2.330708661	2.368
energyConsumptionEmpty	9.130533539	9.655974548
energyConsumptionLoaded	22.28108236	21.92839817
maxQueueLength	51	50
averageQueueLength	6.426010127	6.364018203

Table 2. KPIs of algorithm MaxNrPassengers-B and algorithm MaxWaitSimple-B on "simplesquare" network.

KPI	MaxNrPassengers-B	MaxWaitSimple-B
passengersServed	299	297
passengersNotServed	0	2
maxPassengerWait	1126.2	1062.2
averagePassengerWait	278.2059527	286.127148
averagePassengerTripTime	120.8013514	122.8789116
averagePassengerKm	0.974986392	0.974954126
totalPassengerKm	288.595972	286.6365132
vehicleKmLoaded	125.7433765	127.6927677
vehicleKmEmpty	48.7458273	47.77602881
vehicleKmEmptyRatio	0.279362999	0.272276495
passengersPerLoadedVehicle	2.294573643	2.244274809

energyConsumptionEmpty	8.774248915	8.599685186
energyConsumptionLoaded	22.63380777	22.98469819
maxQueueLength	51	50
averageQueueLength	6.302413281	6.343123156

Table 3. KPIs of algorithm MaxNrPassengers-C and algorithm MaxWaitSimple-C on "simplesquare" network.

KPI	MaxNrPassengers-C	MaxWaitSimple-C
passengersServed	299	294
passengersNotServed	0	5
maxPassengerWait	1124.2	1162.2
averagePassengerWait	282.326354	300.3254229
averagePassengerTripTime	120.3016835	116.2578231
averagePassengerKm	0.974967741	0.974953546
totalPassengerKm	289.565419	286.6363426
vehicleKmLoaded	127.6826935	130.6219785
vehicleKmEmpty	40.96811476	42.91754752
vehicleKmEmptyRatio	0.242916801	0.247307046
passengersPerLoadedVehicle	2.267175573	2.194029851
energyConsumptionEmpty	7.374260656	7.725158553
energyConsumptionLoaded	22.98288482	23.51195613
maxQueueLength	51	51
averageQueueLength	6.301060833	6.198419072

Table 4. KPIs of algorithm MaxNrPassengers and algorithm MaxWaitSimple on "simplesquare (2)" network.

KPI	MaxNrPassengers	MaxWaitSimple
passengersServed	296	294
passengersNotServed	3	5
maxPassengerWait	1372.421525	1655.2
averagePassengerWait	365.5581644	434.5135522
averagePassengerTripTime	149.5687075	151.0831615
averagePassengerKm	1.272470529	1.267103876
totalPassengerKm	374.1063356	368.7272279
vehicleKmLoaded	146.6579283	141.7286907
vehicleKmEmpty	42.19058793	45.63975723

vehicleKmEmptyRatio	0.223409687	0.243582939
passengersPerLoadedVehicle	2.610619469	2.645454545
energyConsumptionEmpty	7.594305828	8.215156302
energyConsumptionLoaded	26.39842709	25.51116433
maxQueueLength	50	51
averageQueueLength	7.393425437	7.649005751

Table 5. KPIs of algorithm MaxNrPassengers-B and algorithm MaxWaitSimple-B on "simplesquare (2)" network.

KPI	MaxNrPassengers-B	MaxWaitSimple-B
passengersServed	297	294
passengersNotServed	2	5
maxPassengerWait	1498.2	1538.2
averagePassengerWait	367.937197	378.3727782
averagePassengerTripTime	146.0027119	149.0191126
averagePassengerKm	1.271444756	1.266746055
totalPassengerKm	375.0762031	371.156594
vehicleKmLoaded	145.1479823	142.6984295
vehicleKmEmpty	43.65006828	43.60976175
vehicleKmEmptyRatio	0.231199783	0.234073239
passengersPerLoadedVehicle	2.619469027	2.63963964
energyConsumptionEmpty	7.85701229	7.849757115
energyConsumptionLoaded	26.12663681	25.68571731
maxQueueLength	51	51
averageQueueLength	7.249603988	7.276257792

Table 6. KPIs of algorithm MaxNrPassengers-C and algorithm MaxWaitSimple-C on "simplesquare (2)" network.

KPI	MaxNrPassengers-C	MaxWaitSimple-C
passengersServed	297	294
passengersNotServed	2	5
maxPassengerWait	1415.2	1389.2
averagePassengerWait	377.202689	347.8273874
averagePassengerTripTime	148.0040816	146.2541096
averagePassengerKm	1.270769195	1.274508248
totalPassengerKm	373.6061434	372.1564084

vehicleKmLoaded	147.6176601	143.6581523
vehicleKmEmpty	36.80181838	40.19061221
vehicleKmEmptyRatio	0.199554942	0.21860692
passengersPerLoadedVehicle	2.587719298	2.625
energyConsumptionEmpty	6.624327308	7.234310198
energyConsumptionLoaded	26.57117881	25.85846741
maxQueueLength	51	51
averageQueueLength	7.540977791	7.255120669

Table 7. KPIs of algorithm MaxNrPassengers and algorithm MaxWaitSimple on "saclay" network.

KPI	MaxNrPassengers	MaxWaitSimple
passengersServed	511	502
passengersNotServed	48	58
maxPassengerWait	1133.636414	1323.984047
averagePassengerWait	242.0809333	271.4885905
averagePassengerTripTime	476.6328018	472.2391011
averagePassengerKm	5.924851242	5.897024203
totalPassengerKm	2601.009695	2624.17577
vehicleKmLoaded	1765.117347	1810.786639
vehicleKmEmpty	629.4546313	630.5019043
vehicleKmEmptyRatio	0.262867284	0.258266032
passengersPerLoadedVehicle	1.597402597	1.569920844
energyConsumptionEmpty	113.3018336	113.4903428
energyConsumptionLoaded	317.7211225	325.941595
maxQueueLength	14	15
averageQueueLength	1.988583833	2.14044346

Table 8. KPIs of algorithm MaxNrPassengers-B and algorithm MaxWaitSimple-B on "saclay" network.

KPI	MaxNrPassengers-B	MaxWaitSimple-B
passengersServed	520	511
passengersNotServed	40	55
maxPassengerWait	869.5094747	1257.472321
averagePassengerWait	234.3017085	262.070645
averagePassengerTripTime	477.0977376	476.0834101

averagePassengerKm	5.919342728	5.940796109
totalPassengerKm	2616.349486	2578.305511
vehicleKmLoaded	1820.862469	1804.245208
vehicleKmEmpty	595.0194619	617.1172835
vehicleKmEmptyRatio	0.246294926	0.25486365
passengersPerLoadedVehicle	1.582697201	1.513089005
energyConsumptionEmpty	107.1035031	111.081111
energyConsumptionLoaded	327.7552443	324.7641374
maxQueueLength	12	13
averageQueueLength	1.883673811	2.095487425

Table 9. KPIs of algorithm MaxNrPassengers-C and algorithm MaxWaitSimple-C on "saclay" network.

KPI	**MaxNrPassengers-C**	**MaxWaitSimple-C**
passengersServed	500	509
passengersNotServed	59	50
maxPassengerWait	1213.814193	807.665222
averagePassengerWait	250.4402091	230.262997
averagePassengerTripTime	468.0441247	474.5640909
averagePassengerKm	5.870923351	5.943063123
totalPassengerKm	2448.175037	2614.947774
vehicleKmLoaded	1799.04947	1858.243198
vehicleKmEmpty	602.5728121	555.9262193
vehicleKmEmptyRatio	0.250902407	0.23027639
passengersPerLoadedVehicle	1.580901857	1.536523929
energyConsumptionEmpty	108.4631062	100.0667195
energyConsumptionLoaded	323.8289046	334.4837756
maxQueueLength	13	12
averageQueueLength	1.99100723	1.854595081

Table 10. KPIs of algorithm MaxNrPassengers and algorithm MaxWaitSimple on "Goteborg" network.

KPI	**MaxNrPassengers**	**MaxWaitSimple**
passengersServed	479	333
passengersNotServed	112	258
maxPassengerWait	2587.732782	2604.069521

averagePassengerWait	410.2924338	620.435319
averagePassengerTripTime	350.9616708	351.7227586
averagePassengerKm	3.71232026	3.677600953
totalPassengerKm	1510.914346	1066.504276
vehicleKmLoaded	590.954049	426.390951
vehicleKmEmpty	882.4269217	1042.979991
vehicleKmEmptyRatio	0.598912935	0.709813949
passengersPerLoadedVehicle	2.417910448	2.288732394
energyConsumptionEmpty	158.8368459	187.7363984
energyConsumptionLoaded	106.3717288	76.75037118
maxQueueLength	12	28
averageQueueLength	2.268316915	2.842148137

Table 11. KPIs of algorithm MaxNrPassengers-B and algorithm MaxWaitSimple-B on "Goteborg" network.

KPI	MaxNrPassengers-B	MaxWaitSimple-B
passengersServed	517	491
passengersNotServed	74	100
maxPassengerWait	2587.732782	1596.139764
averagePassengerWait	304.881723	445.6266493
averagePassengerTripTime	342.0181435	344.0252381
averagePassengerKm	3.671549944	3.661150324
totalPassengerKm	1740.314673	1537.683136
vehicleKmLoaded	694.6161465	610.8298581
vehicleKmEmpty	761.6260854	850.7508375
vehicleKmEmptyRatio	0.523007827	0.582075858
passengersPerLoadedVehicle	2.304166667	2.298076923
energyConsumptionEmpty	137.0926954	153.1351508
energyConsumptionLoaded	125.0309064	109.9493745
maxQueueLength	10	17
averageQueueLength	2.268316915	2.842148137

Table 10. KPIs of algorithm MaxNrPassengers-C and algorithm MaxWaitSimple-C on "Goteborg" network.

KPI	MaxNrPassengers-C	MaxWaitSimple-C
passengersServed	524	489

passengersNotServed	67	102
maxPassengerWait	2587.732782	1617.648836
averagePassengerWait	328.8085169	438.5030996
averagePassengerTripTime	341.4640523	348.9352381
averagePassengerKm	3.662987642	3.690146466
totalPassengerKm	1681.311328	1549.861516
vehicleKmLoaded	690.2576569	586.0572624
vehicleKmEmpty	773.7722116	873.1938763
vehicleKmEmptyRatio	0.528522149	0.59838492
passengersPerLoadedVehicle	2.295833333	2.348258706
energyConsumptionEmpty	139.2789981	157.1748977
energyConsumptionLoaded	124.2463782	105.4903072
maxQueueLength	11	16
averageQueueLength	1.90074637	2.12458983

TRITA – ABE-MBT-24519

Stockholm, Sweden 2024