


A* search algorithm for an optimal investment problem in Vehicle-Sharing system

Ba Luat Le¹[0000-0002-1980-4274], Layla Martin²[0000-0002-1264-0457], Emrah Demir³[0000-0002-4726-2556], and Duc Minh Vu [0000-0001-5882-3868]

¹ ORLab & Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam
21010554@st.phenikaa-uni.edu.vn

 Corresponding author: minh.vuduc@phenikaa-uni.edu.vn

² Department of Industrial Engineering & Eindhoven AI Systems Institute, Eindhoven University of Technology
l.martin@tue.nl

³ Cardiff Business School, Cardiff University
demire@cardiff.ac.uk

Abstract. We study an optimal investment problem that arises in the context of the vehicle-sharing system. Given a set of locations to build stations, we need to determine i) the order of stations to be built and the volume of vehicles to buy in order to get the target state where all stations are built and ii) the number of vehicles to acquire and their allocation in order to maximize the profit returned by operating the system when some or all stations are open. In this problem, the profit (per period of time) when operating open stations is modeled as a linear optimization problem over a set of open stations. Then, with operating capital, the system owner can open new stations. This characteristic makes the transition time needed for opening a new station a set-dependent function, and the optimal investment problem can be seen as a variant of the Traveling Salesman Problem with set-dependent cost. We propose an A* search algorithm to address this particular Traveling Salesman Problem variant. The experiment results show the advantages of the proposed algorithm over the well-known classic Dijkstra algorithm and open new directions for both exact and approximate A* in future work.

Keywords: Autonomous Mobility On-Demand · vehicle-sharing · traveling salesman problem · A* algorithm

1 Introduction

Mobility on demand (MoD) is a rapidly growing market⁴. With the advanced technology of autonomous vehicles, Autonomous Mobility on demand (AMoD) is becoming increasingly popular since it alleviates some operational difficulties of MoD. The global autonomous mobility market is projected to grow from 5 billion USD (in 2019) to 556 billion USD (in 2026)⁵, promising safety (94% of accidents are caused by human factors), increased throughput, improved efficiency, and more affordable services.

⁴ <https://www.alliedmarketresearch.com/mobility-on-demand-market>

⁵ <https://www.alliedmarketresearch.com/autonomous-vehicle-market>

While car manufacturers and major technology firms have the resources to quickly establish an AMoD system, smaller shared mobility operators and public authorities may encounter challenges in securing sufficient upfront capital to launch the service with a sufficient fleet⁶. Consequently, operators first operate in a smaller region, as studied in the literature on optimal service region design, e.g., [7, 9]. As operators accumulate profits, they can gradually acquire more vehicles and expand their active sites. We study this *refinancing* of the AMoD system, where the operator aims to achieve the desired service area and fleet size as quickly as possible.

Research on different issues of an AMoD system, such as operations of vehicle sharing systems, strategic decisions, and regulations and subsidies in vehicle sharing services, can be found in the literature, e.g., [3, 5, 6, 7], the question of what is the optimal investment sequence to build an AMoD has not been addressed yet. In this research, we consider an AMoD with a target service area as well as a current set of open stations. The operator decides on the order in which they open the stations. The more profit they gather, the faster they can open new stations.

In the following sections, we address the above questions and then analyze the performance of our proposed algorithm. To do so, we review papers close to our research in Section 2. Next, we present the problem statement and related formulations in Section 3. Section 4 presents our solution approach based on the A* search algorithm. Numerical experiments and some encouraging results are presented and analyzed in Section 5. Finally, Section 6 concludes and points out further research directions based on the current research.

2 Literature

The research on operations and planning of AMoD systems consists of studies on various research questions. Still, it mainly focuses on optimizing an already-established vehicle-sharing network. Regarding fleet optimization, we can refer to [5, 6, 12] and [10]. George and Xia [6] study a fleet optimization problem over a closed-queue network. This work suggests basic principles for the design of such a system. Nair and Miller-Hooks [13] use the equilibrium network model to find the optimal configuration of a vehicle-sharing network. The solutions to the model explain the correctness of the equilibrium condition, the trade-offs between operator and user objectives, and the insights regarding the installation of services. Freund et al. [5] address how to (re-)allocate dock capacity in vehicle-sharing systems by presenting mathematical formulations and a fast polynomial-time allocation algorithm to compute an optimal solution. Lu et al. [10] consider the problem of allocating vehicles to service zones under uncertain one-way and round-trip rental demand. The authors of recent papers related to the topic [3] present a model that tackles empty-car routing issues in car-sharing systems. Simulation results with real-world data demonstrate the benefits of the model and the routing policy compared with various other approaches. Regarding policies, Martin et al. [11] conclude that using driverless vehicles and human-driven vehicles can improve profits, and operators can gain new unprofitable markets for them. The authors propose a model and an algorithm

⁶ <https://www.weforum.org/agenda/2021/11/trends-driving-the-autonomous-vehicles-industry/>

to find maximum profit while considering driverless and human-driven vehicles. Hao and Martin [7] present a model that studies the impact of regulations on the decisions of vehicle-sharing operators and measures the efficiency and effectiveness of these regulations. The results show that the interdependencies between regulations and societal welfare indicators are non-trivial and possibly counterintuitive. To conclude, we observe that all the research so far has tried to address different questions with the goal of optimizing an already-established vehicle-sharing network. However, the question of how we should establish new stations and acquire new vehicles has not been addressed yet. In the following, we present an optimization problem where we want to determine the order of establishing stations and the fleet size to achieve the final state where all stations are open in the shortest time.

3 Problem Statement and Formulation

We study the best investment strategy for an AMoD (Autonomous Mobility-on-Demand) operator to increase their fleet size and operating area.

The AMoD operator's business area comprises stations, ($\mathcal{R} : \{1, \dots, R\}$). "Station" can also refer to a virtual location, e.g., the center of a region in a free-floating system. Operating station i incurs initial cost c_i^b related to construction, permits, or marketing. Some stations are already open, and profit will be collected from the already-open stations to prepare funds for new stations. The operator incrementally grows the fleet to reach the optimal size promptly while ensuring acceptable service levels within a gradually expanding operating area.

At a given open station i , customers promptly begin their journeys to a different station j . In cases where a station is closed, customers who would have started or finished their journeys there can use a nearby station instead. The customer arrivals follow a Poisson distribution with an arrival rate of λ_{ij} , which depends on the stations' operational status (open or closed). The travel times between stations are exponentially distributed, with an average of $1/\mu_{ij}$, where μ_{ij} denotes the return rate. These arrival and return rates remain constant and are determined solely by whether stations i and j are open.

The operator faces the task of determining the fleet size n at any given time, with the fleet only allowed to grow during the expansion process. Each new vehicle acquisition comes with a procurement cost of c^p . The fleet size must be sufficiently large to serve at least a fraction α of all customers, meeting the minimum service level requirement for the AMoD system. Throughout the development of the AMoD service, it is crucial to keep the service level constant to offset potential learning effects that could deter customers from using the service [4]. To achieve the minimum service level α at all open stations, the operator can rebalance vehicles between stations i and j , incurring a cost of c_{ij}^r . The operator receives a contribution margin of δ_{ij} for each served customer traveling from station i to station j , representing the payoff minus direct operating costs like fuel and periodic repairs.

As a result, the problem involves two interconnected decision-making aspects: first, determining the optimal investment schedule, which entails deciding when and where to open new stations and how many vehicles to procure at each stage, and second,

managing the fleet's operation, which involves making decisions on vehicle rebalancing. In the following, we present our model to determine the optimal fleet and then propose an algorithm to determine the optimal investment sequence.

3.1 Semi-Markov Decision Process for Determining the Optimal Fleet Size

We see the optimal investment scheduling problem of AMoD operators as a Semi-Markov Decision Process (SMDP) because of the nature of the investment problem. In an SMDP, the system's state evolves according to a semi-Markov process, and the decision-maker selects actions based on the current state.

States Each state $s \in \mathcal{S}$ describes the current fleet of size n and the currently opened stations, given by $x_i = 1$ if station $i \in \mathcal{R}$ is open, 0 otherwise.

$$s = \langle n, x_1, \dots, x_R \rangle$$

Each state s is associated with an operational profit $p(s)$ per period, which is calculated by subtracting the rebalancing costs from the contribution margins and an acquisition cost $c(s)$ related to the procurement cost of all vehicles and the cost incurred due to opening station. Apparently, we only need to consider states with positive operational profit in our investment scheme. Regarding this point, the set of states with positive operational profit and the starting state is denoted as \mathcal{S} . Also, if a state s' contains all open stations in a state s , we can easily see and prove that $p(s') \geq p(s)$. For referencing the fleet size and open stations of a specific state s , the notation $n(s)$ and $x_i(s)$ are utilized, respectively. Then, the value of $c(s)$ is determined as follows:

$$c(s) = n(s) \cdot c^p + \sum_{i \in \mathcal{R}} x_i(s) c_i^b$$

Actions Actions refer to the operator's procurement decision, resulting in a state transition into the target state $t \in \mathcal{S}$. Every state $s \in \mathcal{S}$ permits transitions to all other states such that no stations are being closed, i.e., $s \rightarrow t$ exists if $x_i(s) \leq x_i(t) \forall i$.

The time $\tau(s, t)$ necessary for a state transition from state s to a state t depends on the operational profit $p(s)$ and the necessary investment volume $C(s, t)$ where

$$C(s, t) = c(t) - c(s) = (n(t) - n(s)) \cdot c^p + \sum_{i \in \mathcal{R}} (x_i(t) - x_i(s)) \cdot c_i^b.$$

Given that we do not consider partial states (e.g., a state without optimal fleet size) equivalent to $p(s)$ as the maximum profit corresponding to state s , the optimal decision is to transition to the next state as soon as possible. Thus,

$$\tau(s, t) = \frac{C(s, t)}{p(s)}$$

We observe that if $|t| \geq |s| + 2$, it is better to transition to an immediate state s' where $|s| < |s'| < |t|$ since $\frac{C(s, t)}{p(s)} \geq \frac{C(s, s')}{p(s)} + \frac{C(s', t)}{p(s')}$ because $p(s) \leq p(s')$ and $C(s, t) = C(s, s') + C(s', t)$. Therefore, we only need to consider actions between two consecutive states in any optimal investment scheme.

3.2 Mathematical model for calculating optimal profit and minimum acquisition cost

To compute the operational profit $p(s)$ per state $s \in \mathcal{S}$, we formulate the rebalancing problem as an open-queueing network (in line with, e.g., [3, 7, 9, 11]), and optimize over it to maximize operational profits. Given a set of available stations, the model determines the necessary fleet size to reach the service level and rebalance. Since we want to maximize profit and minimize the corresponding acquisition cost, our objective function is hierarchical since we optimize the second objective after minimizing the first objective.

To start, we denote f_{ij}, e_{ij} ($i \neq j$) as the number of occupied and empty vehicles traveling from i to j and e_{ii} as the number of idle vehicles currently parked at station i . To determine the maximal operational profit per period for state s , we solve (1) - (7) for all opening stations in $R_s = \{i \in \mathcal{R} | x_i(s) = 1\}$. The mathematical formulation is expressed as follows:

$$P(obj_1, obj_2) = \left(\max_{\alpha} \alpha \left(\sum_{i \in R_s} \sum_{j \in R_s} \lambda_{ij} \delta_{ij} - \sum_{i \in R_s} \sum_{j \in R_s} c_{ij}^r \mu_{ij} e_{ij} \right), \min \left(n \cdot c^p + \sum_{i \in R_s} c_i^b \right) \right) \quad (1)$$

subject to

$$\lambda_{ij} = \mu_{ij} f_{ij}, \quad \forall i, j \in R_s \quad (2)$$

$$\sum_{j \in R_s \setminus \{i\}} \mu_{ji} e_{ji} \leq \sum_{j \in R_s \setminus \{i\}} \lambda_{ij}, \quad \forall i \in R_s \quad (3)$$

$$\sum_{j \in R_s} \lambda_{ij} + \sum_{j \in R_s} \mu_{ij} e_{ij} = \sum_{j \in R_s} \mu_{ji} e_{ji} + \sum_{j \in R_s} \lambda_{ji}, \quad \forall i \in R_s \quad (4)$$

$$\frac{\alpha}{1-\alpha} \leq e_{ii}, \quad \forall i \in R_s \quad (5)$$

$$\sum_{i, j \in R_s} (e_{ij} + f_{ij}) = n, \quad (6)$$

$$e_{ij}, f_{ij} \geq 0, \quad \forall i, j \in R_s \quad (7)$$

The objective function (1) maximizes the profit, which consists of the contribution margin of all served customers minus rebalancing costs, multiplied by the availability α to account for those customers who cannot be served, and then minimizes the corresponding set-up fee. Constraints (2)-(4) are linearizations of the flow constraints in queueing networks and almost directly follow from [3]. Unlike [3], we require that the system achieve a service level of at least α and thus remove any notion of an upper bound on demand. Constraints (5) set the required ‘‘safety stock’’, i.e., the number of vehicles that must remain at station i in the steady state to fulfill at least a fraction α of the demand. These constraints follow the fixed population mean approximation in open queueing networks due to [14]. Constraint (6) bounds the fleet size, and constraints (7) define the domain.

4 Solution Approach

Remember that in our problem, the optimal time for opening a new station depends on the profit of already open stations, or it is a set-dependent cost. Because the number of such sets is exponential in terms of the number of stations, a mathematical formulation may require too many variables and constraints; therefore, it seems impractical to model and solve the corresponding formulation using state-of-the-art solvers.

We can consider our investment problem as a variant of the well-known Traveling Salesman Problem (TSP) with set-dependent travel costs. Actually, considering a permutation (u_1, u_2, \dots, u_n) that presents an order, we open stations. Each subpath (u_1, u_2, \dots, u_i) is mapped to a state s_i where $x_k(s_i) = 1$ if $u_j = k$ for some $j = 1..i$. The cost between two consecutive states, s_i and s_{i+1} , is calculated by the formulations in Section 3.1, which depends on the set of open stations in s_i . In other words, it is a set-dependent cost function. While there is much research for TSP in general and several research studies on order-dependent travel cost TSP [1, 2] in particular, our work seems the first of this kind. In the subsequent sections, we will present part of our attempt to address this challenging question.

4.1 Heuristic strategy for A* algorithm

We can model our investment problem as a shortest-path problem. Let us consider a graph $G = (V, A)$ where each node $n_s \in V$ corresponds to the state s . Each arc $(n_s, n_{s'}) \in A$ corresponds to a feasible action between two consecutive states s and s' with cost $C(s, s')$. Finding the shortest investment time is equivalent to finding the shortest path from the node n_{s_0} to the node n_{s_f} where s_0 and s_f are the initial state and the final state, respectively. Since we can define a 1-1 mapping between s and n_s , we use s instead of n_s subsequently to simplify the notation.

To solve this shortest path problem, we rely on the A* algorithm [8]. Given a state s , unlike the classic Dijkstra algorithm, which only evaluates the cost of the shortest path $g(s)$ from the source s_0 to s , A* also evaluates the cost $h(s)$ from s to the final state s_f , and the cost for each node s is then $f(s) = g(s) + h(s)$ instead of $g(s)$. The algorithm A* can always find the shortest path from s_0 to s_f if $h(s)$ does not exceed the cost of the shortest path from s to s_f for any s (admissible property [8]). Otherwise, A* becomes a heuristic algorithm. Due to the space, we present our preliminary results of both exact and approximate A* algorithms we developed for the problem.

Simple heuristic for A* We start with some of the simplest heuristics for A*. Given the current, next, and final states be s, s' , and s_f , the cost of the shortest path from n_0 to s' , $g(s')$, is $g(s') = g(s) + \frac{c(s') - c(s)}{p(s)}$. Several simple ways to calculate $h(s')$ are as follows (where eh and ah denote exact and approximate heuristics, respectively):

$$eh_1(s') = \frac{c(s_f) - c(s')}{P_{R-1}} \quad (8)$$

$$ah_1(s') = \frac{c(s_f) - c(s')}{p(s')} \quad (9)$$

Heuristic functions (8), (9) underestimate and overestimate the shortest time of the optimal path from s_0 to s_f that passes through s' . Here, P_{R-1} denotes the maximum profit for any state having $R-1$ open stations. Using a linear combination, we obtain other heuristics where $\gamma \in [0, 1]$ is a parameter that can be a fixed constant or dynamically adjusted during the execution of the algorithm. We aim to test whether we can obtain simple heuristics that may not be optimal but can quickly find reasonable solutions.

$$ah_2(s') = \gamma eh_1(s') + (1 - \gamma)ah_1(s') \quad (10)$$

Stronger lower bound heuristics for A* Now assume $s = s_1$ is the current state. Let s_1, s_2, \dots, s_k be a sequence of states where s_{i+1} is obtained from s_i by adding a new station and $s_k = s_f$ be the final state where all stations are open. The total transition time from state s_1 to state s_k , $\tau(s_1, \dots, s_k)$, is:

$$\tau(s_1, \dots, s_k) = \frac{c(s_2) - c(s_1)}{p(s_1)} + \frac{c(s_3) - c(s_2)}{p(s_2)} + \dots + \frac{c(s_k) - c(s_{k-1})}{p(s_{k-1})} \quad (11)$$

We denote $\underline{\Delta}_c(s_i)$ as a lower bound of the difference of the acquisition cost $c(s_{i+1}) - c(s_i)$ between two consecutive states s_i and s_{i+1} . Let P_m be a state with the maximum profit among all states with m opening stations. We can find the value of P_m by solving the model (22) - (34) (see Appendix), which aims to maximize the profit and minimize the corresponding acquisition cost given a fixed number of stations that can be opened. Then, we obtain $p(s_i) \leq P_{|s_i|}, \forall i = 1, \dots, k$. The values of $P_{|s_i|}$ define an increasing sequence since we open more stations. Therefore, we have $p(s_i) \leq P_{|s_i|} \leq P_{|s_{k-1}|} = P_{R-1}, \forall i = 1, \dots, k-1$. Given that $c(s_{i+1}) - c(s_i) \geq \underline{\Delta}_c(s_i)$ or $c(s_{i+1}) - c(s_i) - \underline{\Delta}_c(s_i) \geq 0$, therefore, $\forall i = 1, \dots, k-1$ we have:

$$\frac{c(s_{i+1}) - c(s_i)}{p(s_i)} = \frac{\underline{\Delta}_c(s_i) + (c(s_{i+1}) - c(s_i) - \underline{\Delta}_c(s_i))}{p(s_i)} \quad (12)$$

$$= \frac{\underline{\Delta}_c(s_i)}{p(s_i)} + \frac{c(s_{i+1}) - c(s_i) - \underline{\Delta}_c(s_i)}{p(s_i)} \quad (13)$$

$$\geq \frac{\underline{\Delta}_c(s_i)}{P_{|s_i|}} + \frac{c(s_{i+1}) - c(s_i) - \underline{\Delta}_c(s_i)}{P_{R-1}} \quad (14)$$

and consequently

$$\tau(s_1, \dots, s_k) \geq \left(\sum_{i=1}^{k-1} \frac{\underline{\Delta}_c(s_i)}{P_{|s_i|}} \right) + \frac{c(s_k) - c(s_1) - \sum_{i=1}^{k-1} \underline{\Delta}_c(s_i)}{P_{R-1}} \quad (15)$$

Inequality (15) gives us a more robust lower bound than the simple one presented in (8).

Evaluate the lower bound $\underline{\Delta}_c(s_1)$ From (6), we see that with each state S , the optimal number of vehicles n is equal to $\sum_{i,j \in S} (e_{ij} + f_{ij})$. Following *obj2*, when we open a new station, the acquisition cost includes the cost of station setup and the new vehicle acquisition cost. We assume that the difference in acquisition cost between two consecutive states depends on the values f_{ij}, e_{ij} of the new station i . With this assumption,

the minimum acquisition cost of opening station i from a given state S ($i \notin S$) to obtain maximum profit is determined by $\Delta_c(S, i)$. In other words, $\Delta_c(S, i)$ presents the lower difference in acquisition cost between two consecutive states in which the next state is reached by opening station i from the state S .

$$\Delta_c(S, i) = \sum_{j \in S} (e_{ij} + e_{ji} + f_{ij} + f_{ji})c^P + c_i^b \quad \forall i \notin S \quad (16)$$

Then, $c(s_{i+1}) - c(s_i) \geq \min_{o \notin s_i} \Delta_c(s_i, o) \quad \forall i = 1, 2, \dots, k-1$.

Next, we indicate how to obtain the lower bound $\Delta_c(s_i, o)$ using equations (16). Assuming $T \subset \mathcal{R}$ be the state with any t stations not in S , $t < R - |S|$, $S \cap T = \emptyset$. Let $o \notin S \cup T$, and we evaluate $\Delta C(S, t, i)$ - the minimum difference acquisition cost when building a new station i starts from state $S \cup T$ with any state T such that $|T| = t$.

Underestimate acquisition cost We rewrite $\Delta C(S, t, i)$ using equation (16) as follows:

$$\Delta C(S, t, i) = \min_{T \subset \mathcal{R}, |T|=t} \sum_{j \in S \cup T} (e_{ij} + e_{ji} + f_{ij} + f_{ji})c^P + c_i^b \quad (17)$$

and since e_{ij}, e_{ji} and c^P are non-negative, we have

$$\Delta C(S, t, i) \geq \min_{T \subset \mathcal{R}, |T|=t} \sum_{j \in S \cup T} (f_{ij} + f_{ji})c^P + c_i^b \quad (18)$$

Since $c_i^b + \sum_{j \in S} (f_{ij} + f_{ji})c^P$ is a constant, we will develop a lower bound for the sum $\sum_{j \in T} (f_{ij} + f_{ji})c^P$. Apparently, $\sum_{j \in T} (f_{ij} + f_{ji})c^P$ cannot be smaller than the sum of $|T|$ smallest values of $(f_{ij} + f_{ji})c^P$ where $j \notin S \cup \{i\}$. Therefore, we developed the Algorithm 1 to evaluate a lower bound of $\Delta C(S, t, i)$.

Algorithm 1 Lower bound evaluation of acquisition cost

Require: S, i, t

Ensure: A lower bound of $\Delta C(S, t, i)$

- 1: Let $\alpha \leftarrow c_i^b + \sum_{j \in S} (f_{ij} + f_{ji})c^P$
 - 2: Sort $(f_{ij} + f_{ji})_{j \in \mathcal{R} \setminus (S \cup \{i\})}$ increasingly.
 - 3: Let $(f_{i_{j_1}} + f_{j_1 i}) \leq (f_{i_{j_2}} + f_{j_2 i}) \leq \dots \leq (f_{i_{j_k}} + f_{j_k i}) \leq \dots$ be the array after sorting.
 - 4: Let $\beta \leftarrow \sum_{k=1}^t (f_{i_{j_k}} + f_{j_k i})c^P$
 - 5: **Return** $\alpha + \beta$
-

Using Algorithm 1, we have that

$$\Delta_c(s_i, o) \geq \Delta C(s_1, i-1, o) \geq c_o^b + \sum_{j \in s_1} (f_{oj} + f_{jo})c^P + \sum_{k=1}^{i-1} (f_{oj_k} + f_{j_k o})c^P \quad (19)$$

and consequently,

$$c(s_{i+1}) - c(s_i) \geq \min_{o \notin s_i} \Delta_c(s_i, o) \quad (20)$$

$$\geq \min_{o \notin s_i} \left(c_o^b + \sum_{j \in s_1} (f_{oj} + f_{jo})c^P + \sum_{k=1}^{i-1} (f_{ok} + f_{ko})c^P \right) \quad (21)$$

Use $\underline{\Delta}_c(s_i) = \min_{o \notin s_i} \left(c_o^b + \sum_{j \in s_1} (f_{oj} + f_{jo})c^P + \sum_{k=1}^{i-1} (f_{ok} + f_{ko})c^P \right)$ in inequality (15), we obtain a lower bound heuristic for A*, called eh_2 , which is stronger than the simple one eh_1 . However, we need to solve it online.

We obtain a weaker lower bound version of eh_2 by fixing s_1 , e.g., to the initial state s_0 . Still, this strategy may reduce the total running time since the value of $\underline{\Delta}_c(s_i)$ needs to be calculated only once, while with eh_2 , we will calculate $\underline{\Delta}_c(s_i)$ for each extracted state $s = s_1$ from the queue. Let eh_3 be this lower bound heuristic.

5 Numerical Experiments

In this section, we present the numerical design and then report and compare the experiment results of exact and approximate algorithms in finding an optimal schedule investment. The algorithm and the formulations were written in C++, and the MILP models were solved by CPLEX 22.1.1. The experiments were run on an AMD Ryzen 3 3100 machine with a 4-core processor, 3.59 GHz, and 16GB of RAM on a 64-bit Windows operating system.

5.1 Numerical Design

We conducted experiments on randomly generated datasets, following a similar approach as Martin et al. [7]. To model the real-world transportation network structure, our datasets vary in size ($R \in \{7, 9, 16, 19, 25\}$) and geographic distributions of station locations, including circular (C), hexagonal (H), and quadratic (Q) layouts.

We randomly sample hourly arrival rates from a uniform distribution $\mathcal{U}[80, 120]$ (BAL). We generate additional instances for larger datasets to reflect imbalances (IMB) in arrival rates. Arrival rates are higher near the city center ($\mathcal{U}[110, 140]$) and lower in suburban areas ($\mathcal{U}[60, 90]$). There are a total of 11 instances, which include 7 balance instances and 4 imbalance instances. Across all instances, customers travel to other stations with equal probability. Vehicles require a time of $t_{ij} = 3/60 + d_{ij}/25$ hours for the trip from station i to station j , where d_{ij} represents the Euclidean distance between the two stations. The minimum service level for customer retention is $\alpha = 0.5$. Rebalancing a vehicle costs \$0.3 per kilometer. Transporting a customer between two locations yields a contribution of \$0.3 per kilometer, representing revenues of approximately \$1 per kilometer minus direct costs. Procurement costs (c^P) amount to \$1. The operating cost for each opening station is randomly generated from $\mathcal{U}[1000, 3000]$.

The investment starts with a set of initially open stations. We assume that initially, there is a budget of $B = 10000$, which is optimally utilized to construct the initial stations to maximize the initial profit. With smaller instances (less than 10 stations), we

use a dynamic budget of $500 \times R$ to avoid opening too many stations at the initial state. We simulate this process through formulations similar to (22) - (34) (see Appendix). With this budget, the initial state has 5 – 7 open stations for larger instances and 2 – 3 for instances with less than 10 stations. Then, the A* algorithm will find an optimal investment plan starting from the initial state with a certain number of already opened stations obtained from the formulations.

5.2 Experiment Results

In the following, we assess two points:

1. We compare the performance of the exact A* heuristics and Dijkstra algorithm based on the execution time, the number of states explored, and the number of states remaining in the priority queue.
2. We compare the performance of approximate A* heuristics in terms of optimal gap, and execution time.

Table 1 analyzes the performance of exact A* algorithms and the Dijkstra algorithm by reporting their running time in seconds (column Time (s)), the number of nodes extracted by the A* algorithm (column Exp.), and the number of nodes still in the queue (column Rem.) with the optimal value (column Opt.) obtained from all exact algorithms.

Table 1. Results of exact A* heuristic and Dijkstra algorithms

Instance	Opt.	Dijkstra	A* + eh_1		A* + eh_2			A* + eh_3			
		Time (s)	Exp.	Rem.	Time (s)	Exp.	Rem.	Time (s)	Exp.	Rem.	Time (s)
C-7-BAL	1563.19	<1	17	144	<1	12	10	<1	12	12	<1
H-7-BAL	1524.71	<1	17	13	<1	11	9	<1	12	12	<1
Q-9-BAL	435.53	<1	33	31	<1	14	25	<1	20	25	<1
Q-16-BAL	420.87	1	392	173	1	227	236	1	243	231	1
Q-16-IMB	723.69	1	340	176	1	150	208	1	170	195	1
C-19-BAL	1054.83	14	2733	2060	12	1453	2234	9	1603	2156	9
C-19-IMB	1681.48	14	2292	1473	11	902	1421	7	1103	1378	7
H-19-BAL	1028.32	13	2303	1710	11	903	1568	7	1149	1656	8
H-19-IMB	1833.02	14	1812	1720	10	592	1299	5	888	1385	6
Q-25-BAL	711.31	1313	140878	119407	1032	35068	84174	452	44551	90771	508
Q-25-IMB	1185.37	1311	124532	105743	978	18440	55015	328	24975	61453	372

The experiments show that the time required to open all stations is significantly longer for datasets with imbalanced arrival rates than for balanced datasets due to decreased profit margins. Since eh_2 is the strongest lower bound heuristic, we see that the corresponding running time, number of expanded nodes, and number of remaining nodes are the smallest among all the exact methods. Table 1 indicates that utilizing the A* algorithm with eh_2 has significantly reduced both the computation time and

the number of vertices explored compared to underestimating the shortest time of the optimal path eh_1 . The exact heuristic eh_3 is also quite effective, providing computational stability without the need for online updates for each node.

We can also observe that the number of visited vertices and execution time increase exponentially as the number of stations increases. Therefore, we experimented with several heuristic approximation approaches in the A* search algorithm to quickly find a sufficiently good investment schedule. Table 2 reports the performance of approximate A* search algorithms with respect to different values of γ . Apparently, the larger value of γ leads to better objective values and longer running time. While the heuristic approximation approaches achieve excellent time efficiency and relatively small gaps, they are highly dependent on the data. When we modify the range of the values of some parameters within the instances, these approaches become less effective. However, they remain valuable when dealing with datasets with a large number of stations.

Table 2. Non-bounded approximation algorithms with simple heuristics

Instance	Opt.	A* + ah_1		A* + $ah_2(\gamma = 0.3)$		A* + $ah_2(\gamma = 0.5)$		A* + $ah_2(\gamma = 0.7)$	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
C-7-BAL	1563.19	9.01	<1	4.34	<1	0.00	<1	0.00	<1
H-7-BAL	1524.71	8.85	<1	4.37	<1	4.37	<1	0.00	<1
Q-9-BAL	435.53	4.63	<1	1.29	<1	1.29	<1	0.00	<1
Q-16-BAL	420.87	7.18	<1	3.86	<1	1.78	<1	0.00	1
Q-16-IMB	723.69	6.00	<1	3.82	<1	1.47	<1	0.00	1
C-19-BAL	1054.83	12.76	<1	7.78	<1	0.94	<1	0.00	3
C-19-IMB	1681.48	17.25	<1	9.62	<1	0.33	<1	0.00	3
H-19-BAL	1028.32	6.40	1	2.11	<1	1.20	<1	0.00	2
H-19-IMB	1833.02	11.98	<1	9.19	<1	4.48	<1	0.00	1
Q-25-BAL	711.31	15.10	1	10.08	1	6.84	1	0.53	39
Q-25-IMB	1185.37	11.87	1	7.28	1	3.39	1	1.26	15

Finally, we report in Table 3 the performance of weighted A* variants where we multiply the values of eh_2 and eh_3 by either 1.05 or 1.1. The best solutions returned by this variant ensure that the gap between the optimal solutions and those best solutions is at most 5% or 10%, respectively. While this heuristic is slower than the ones mentioned in Table 2, it ensures the optimal gap that the approximate heuristics mentioned in Table 2 can not. Also, it is several times faster than the exact heuristic. Results in Table 3 indicate that the optimal gap obtained by those approximation algorithms is very small (less than 1% for all instances). This again highlights the effectiveness of both heuristics eh_2 and eh_3 .

To conclude the section, we observe that for those benchmark instances, exact methods can provide optimal solutions in a reasonable amount of time for those benchmark instances. The proposed lower bound heuristic eh_2 beats simple heuristic eh_1 and the Dijkstra algorithm. The simple approximate A* heuristic can give quite good results with a small computation time, while the weighted A* heuristic based on the best lower

Table 3. Bounded approximation algorithms based on stronger lower-bound heuristic

Instance	Opt.	$A^* + 1.1 * eh_2$		$A^* + 1.1 * eh_3$		$A^* + 1.05 * eh_2$		$A^* + 1.05 * eh_3$	
		Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
C-7-BAL	1563.19	0.00	<1	0.00	<1	0.00	<1	0.00	<1
H-7-BAL	1524.71	0.00	<1	0.00	<1	0.00	<1	0.00	<1
Q-9-BAL	435.53	0.67	<1	0.67	<1	0.00	<1	0.00	<1
Q-16-BAL	420.87	0.17	1	0.13	1	0.13	1	0.13	1
Q-16-IMB	723.69	0.73	1	0.00	1	0.00	1	0.00	1
C-19-BAL	1054.83	0.40	6	0.09	6	0.09	7	0.09	8
C-19-IMB	1681.48	0.10	4	0.04	5	0.04	5	0.04	6
H-19-BAL	1028.32	0.24	3	0.07	4	0.07	5	0.01	6
H-19-IMB	1833.02	0.27	4	0.07	4	0.14	4	0.00	5
Q-25-BAL	711.31	0.26	194	0.08	241	0.04	306	0.03	370
Q-25-IMB	1185.37	0.43	117	0.09	135	0.10	181	0.00	241

bound heuristic can reduce the computation time by about half and maintain a small optimal gap.

6 Conclusion

In this research, we study an investment problem that arises in the context of Autonomous Mobility on-Demand systems. Given some already open stations, the question is to determine the optimal order of opening the remaining stations to minimize the total opening time. We model this investment problem as a Semi-Markov Decision Process and view this problem as a variant of the TSP problem, where the cost between two vertices s and t is dependent on the set of already visited vertices belonging to the path from the source vertex to vertex s . This special cost function makes the problem impossible to model and solve with current mixed integer solver technology. We then develop and solve this new variant using the A^* algorithm. The experiment results show that the A^* algorithm can reduce by half the running time of the Dijkstra algorithm and a simple, exact A^* algorithm. Regarding the approximate A^* search, the result shows that we can obtain reasonable solutions with a small computation effort.

It is still a challenging task to solve larger problems. Therefore, we are developing and testing more robust lower-bound heuristics for exact A^* search. Also, we are testing new approximate heuristics for A^* search that take ideas from the lower bound heuristics. The initial results show that we can solve larger instances in a shorter time using both methods. Also, the approximate A^* heuristic gives similar results to those returned by the exact A^* heuristic in many instances.

References

- [1] Alkaya, A.F., Duman, E.: Combining and solving sequence dependent traveling salesman and quadratic assignment problems in pcb assembly. *Discrete Applied Mathematics* **192**, 2–16 (2015)

- [2] Bigras, L.P., Gamache, M., Savard, G.: The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Optimization* **5**(4), 685–699 (2008)
- [3] Braverman, A., Dai, J.G., Liu, X., Ying, L.: Empty-car routing in ridesharing systems. *Operations Research* **67**(5), 1437–1452 (2019)
- [4] DeCroix, G., Long, X., Tong, J.: How service quality variability hurts revenue when customers learn: Implications for dynamic personalized pricing. *Operations Research* (2021)
- [5] Freund, D., Henderson, S.G., Shmoys, D.B.: Minimizing multimodular functions and allocating capacity in bike-sharing systems. *Production and Operations Management* **27**(12), 2346–2349 (2018)
- [6] George, D.K., Xia, C.H.: Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research* **211**(1), 198–207 (2011)
- [7] Hao, W., Martin, L.: Prohibiting cherry-picking: Regulating vehicle sharing services who determine fleet and service structure. *Transportation Research Part E: Logistics and Transportation Review* **161**, 102692 (2022)
- [8] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* **4**(2), 100–107 (1968). <https://doi.org/10.1109/TSSC.1968.300136>
- [9] He, L., Mak, H.Y., Rong, Y., Shen, Z.J.M.: Service region design for urban electric vehicle sharing systems. *Manufacturing & Service Operations Management* **19**(2), 309–327 (2017)
- [10] Lu, M., Chen, Z., Shen, S.: Optimizing the profitability and quality of service in carshare systems under demand uncertainty. *Manufacturing & Service Operations Management* **20**(2), 162–180 (2018)
- [11] Martin, L., Minner, S., Pavone, M., Schiffer, M.: It’s all in the mix: Technology choice between driverless and human-driven vehicles in sharing systems (2021), available at SSRN 4190991
- [12] Nair, R., Miller-Hooks, E.: Fleet management for vehicle sharing operations. *Transportation Science* **45**(4), 524–540 (2011)
- [13] Nair, R., Miller-Hooks, E.: Equilibrium network design of shared-vehicle systems. *European Journal of Operational Research* **235**(1), 47 – 61 (2014)
- [14] Whitt, W.: Open and closed models for networks of queues. *AT&T Bell Laboratories Technical Journal* **63**(9), 1911–1979 (1984). <https://doi.org/10.1002/j.1538-7305.1984.tb00084.x>

Appendix

Evaluate profit’s upper bound for any state opening m stations

Assuming that we can open m stations, the following mixed integer formulation (22) - (34). will help us determine the set of stations to open and the number of vehicles to acquire to maximize the profit while minimizing the corresponding acquisition cost.

$$P(obj_3, obj_4) = \left(\max \alpha \left(\sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{R}} \lambda_{ij} \delta_{ij} - \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{R}} c_{ij}^r \mu_{ij} e_{ij} \right), \min n \cdot c^p + \sum_{i \in \mathcal{R}} c_i^b y_i \right) \quad (22)$$

subject to

$$\mu_{ij} f_{ij} = \lambda_{ij} x_{ij}, \quad \forall i, j \in \mathcal{R} \quad (23)$$

$$\sum_{j \in \mathcal{R} \setminus \{i\}} \mu_{ji} e_{ji} \leq \sum_{j \in \mathcal{R} \setminus \{i\}} \lambda_{ij} x_{ij}, \quad \forall i \in \mathcal{R} \quad (24)$$

$$\sum_{j \in \mathcal{R}} \lambda_{ij} x_{ij} + \sum_{j \in \mathcal{R}} \mu_{ij} e_{ij} = \sum_{j \in \mathcal{R}} \mu_{ji} e_{ji} + \sum_{j \in \mathcal{R}} \lambda_{ji} x_{ji}, \quad \forall i \in \mathcal{R} \quad (25)$$

$$\frac{\alpha}{1-\alpha} \cdot y_i \leq e_{ii}, \quad \forall i \in \mathcal{R} \quad (26)$$

$$\sum_{i, j \in \mathcal{R}} (e_{ij} + f_{ij}) = n, \quad (27)$$

$$e_{ij} + f_{ij} \leq M x_{ij}, \quad \forall i, j \in \mathcal{R} \quad (28)$$

$$x_{ij} \leq y_i, \quad \forall i, j \in \mathcal{R} \quad (29)$$

$$x_{ij} \leq y_j, \quad \forall i, j \in \mathcal{R} \quad (30)$$

$$y_i + y_j - x_{ij} \leq 1 \quad \forall i, j \in \mathcal{R} \quad (31)$$

$$\sum_{i \in \mathcal{R}} y_i = m \quad (32)$$

$$e_{ij}, f_{ij} \geq 0, \quad \forall i, j \in \mathcal{R} \quad (33)$$

$$x_{ij}, y_i \in \{0, 1\}, \quad \forall i, j \in \mathcal{R} \quad (34)$$

Strategy for generating initial state given budget B

Given the budget B , the budget is optimally allocated to construct the initial state. The budget limits the number of stations and initial vehicles that can be procured initially, ensuring that the system achieves a predetermined initial profit for development purposes. We can obtain the optimal investment for the initial state using the formulations (22) - (34) and substitute the constraints (32) by the budget constraints (35) as follow:

$$n \cdot c^p + \sum_{i \in \mathcal{R}} c_i^b y_i \leq B, \quad (35)$$