# A Modified Artificial Bee Colony Algorithm for the Dynamic Ride-hailing Sharing Problem

**Xingbin Zhan**
Department of Civil Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: xbzhan@hku.hk

**W. Y. Szeto**, Ph.D.
Department of Civil Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: ceszeto@hku.hk

**C. S. Shui**, Ph.D.
Department of Transportation and Logistics Management
National Chiao Tung University
Hsinchu, Taiwan
Email: csshui@nctu.edu.tw

**Xiqun (Michael) Chen**, Ph.D.
College of Civil Engineering and Architecture
Zhejiang University
Hangzhou 310058, China
Email: chenxiqun@zju.edu.cn

**Acknowledgments**
This research is jointly supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region of China (HKU 17201217) and a grant from the University Research Committee of the University of Hong Kong (201811159080). The fourth author Xiqun Chen is financially supported by the National Key Research and Development Program of China (2018YFB1600900), the National Natural Science Foundation of China (71771198, 71922019), the joint project of the National Natural Science Foundation of China and Joint Programming Initiative Urban Europe (NSFC – JPI UE) ('U-PASS', 71961137005), and the Zhejiang Provincial Natural Science Foundation of China (LR17E080002). We would like to thank Didi to provide ride-hailing data.

# A Modified Artificial Bee Colony Algorithm for the Dynamic Ride-hailing Sharing Problem


## Abstract

Ride-hailing sharing involves grouping ride-hailing customers with similar trips and time schedules to share the same ride-hailing vehicle to reduce their total travel cost. With the current information and communication technology, ride-hailing customers and drivers can be matched in real-time via a ride-hailing platform. This paper formulates a dynamic ride-hailing sharing problem that simultaneously maximizes the number of served customers, minimizes the travel cost and travel time ratios, and considers the capacity, time window, and travel cost constraints. The travel cost ratio is the ratio of actual passengers' fare to the passengers' fare without ride-hailing sharing, whereas the travel time ratio is defined as the actual travel time (including waiting time) over the maximum allowable travel time. To solve the dynamic problem, it is divided into many small and continuous static subproblems with an equal time interval. Each subproblem is solved by a modified artificial bee colony (MABC) algorithm with path relinking, while the contraction hierarchies and vantage point tree are used to determine the shortest path and accelerate the algorithm, respectively. Problem properties and the performance of the proposed solution method are demonstrated using large-scale real-time data from Didi that is the largest ride-hailing company in China. The proposed method is shown to outperform the benchmark, i.e., greedy randomized adaptive search procedure (GRASP) with path relinking. The proposed method also performs better when the length of each time interval is longer, and the tolerance for the incremental travel time caused by detours is higher. We also demonstrate that (a) considering both travel cost and travel time ratios in the objective can achieve a better sharing percentage, and balance the increase in the travel time ratio and the decrease in the travel cost ratio compared with the objective that misses either travel time or the travel cost ratio; and (b) the passengers can gain a large out-of-pocket cost saving in the case of ride-hailing sharing while enduring a relatively small increase in travel time compared with the case without ride-hailing sharing.



*Keywords*: Dynamic ride-hailing sharing; artificial bee colony algorithm; path relinking; vantage point tree.


## 1. Introduction

With the development of social economy and motorization, increasing traffic congestion in urban road networks, finite oil supplies, and environmental pollution have aroused great attention from the public. According to Jensen (2005) and Santos et al. (2011), the private car occupancy rates (the number of travelers per vehicle) are quite low in both Europe and the US, reaching 1.8 persons per vehicle for leisure trips and 1.1 for commuting trips. The low occupancy rate has led to a huge waste of social resources. The annual cost of wasted time and fuel caused by traffic congestion in the US was approximately 160 billion dollars in 2015 (Schrank et al., 2015). Moreover, greenhouse gases emitted by vehicles have increased more than double since 1970, while the annual greenhouse emissions were still growing (IPCC, 2015). Another issue is that more travelers choose ride-hailing services (e.g., Didi, Uber, and Lyft) for convenience with an increase in income. The supply of ride-hailing vehicles usually does not meet the travel demand during peak hours, and consequently, travelers have to wait for a long time before using the services or abandon the services and shift to other modes.

One solution to the above problems is ride-hailing sharing. Ride-hailing sharing is a type of ride-hailing service that groups the customers with similar trips and time schedules to share the

same ride-hailing vehicle, which can consequently reduce the total driving distance and fuel cost of the vehicles and increase the vehicle occupancy rate. Ride-hailing sharing services have been provided by private companies such as Didi, Uber, Lyft, etc. in recent years and proved that both the passengers and drivers could benefit from these services. The passengers with loose travel time windows can receive compensation as a return of the increase in travel time, while ride-hailing drivers can serve more passengers and earn more during their available working time. As a real-time service that connects multiple passengers and ride-hailing drivers, the operations of ride-hailing sharing require a third-party platform to provide technical support, including collecting the travel information of ride-hailing drivers and customers (e.g., current locations and customers' preferences) and matching the requests of ride-hailing customers with the vehicles. With the increasing smartphone penetration rate and the development of wireless communication technology, both ride-hailing drivers and customers can access information timely and accurately to implement ride-hailing sharing.

Dynamic ride-hailing sharing problems with different objectives have been studied in the literature. The commonly adopted objectives include maximizing the number of served customers and minimizing the travel time (distance or delay). Moreover, passengers' travel cost is a critical measure when providing ride-hailing sharing services as it often influences whether travelers choose to share rides or not. Santos and Xavier (2015) formulated it in the form of the travel cost ratio, which is the ratio of actual passengers' travel cost to the passengers' travel cost without ride-hailing sharing. However, using the travel cost ratio in the objective function does not prevent the travel time increment due to the detour caused by ride-hailing sharing to be acceptable by passengers. Therefore, we formulate a new dynamic ride-hailing sharing problem that simultaneously maximizes the number of served customers, and minimizes the travel cost and travel time ratios, where the travel time ratio is defined as the actual travel time (including waiting time) over the maximum allowable travel time. To solve the dynamic problem, it is divided into many small and continuous static subproblems with an equal time interval. Each subproblem is solved by a modified artificial bee colony (MABC) algorithm with path relinking, while the contraction hierarchies and vantage point tree are used to determine the shortest path and accelerate the algorithm, respectively. Problem properties and the performance of the proposed solution method are demonstrated using large-scale real-time data from Didi.

The main contributions of this paper can be summarized as follows:

- We present a novel dynamic ride-hailing sharing problem that simultaneously maximizes the weighted number of served customers, minimizes the weighted sum of travel cost and travel time ratios, and considers the constraints of capacity, time window, and travel cost. This problem considers both the travel cost and travel time in the objective function, since two of the most important factors that affect taking ride-hailing sharing service is the cost and time.

- We divide the problem into many small and continuous static subproblems with an equal time interval. We propose a new solution method based on the MABC algorithm to solve the subproblem. To accelerate the solution search, we use the vantage-point (VP) tree to narrow the solution search space of each request by identifying the ride-hailing vehicles near the pickup point of the passengers within a pre-defined radius. The overall solution approach is proved to be efficient in solving large-scale ride-hailing sharing problems.

- Based on real ride-hailing data, we illustrate the performance of the proposed solution approach and show that our approach is more effective than the existing solution method proposed by Santos and Xavier (2015) for the dynamic ride-hailing sharing problem.

The remainder of this paper is organized as follows. In Section 2, we provide an in-depth literature review to show the research gaps. In Section 3, we describe the dynamic ride-hailing sharing problem and formulate the static subproblem. Section 4 proposes the MABC algorithm. Section 5 presents the computational results. Finally, Section 6 concludes the paper and provides an outlook on future research.

## 2. Literature Review

In the literature, the ride-hailing sharing problem can be regarded as a variant of the dial-a-ride problem (DARP). The DARP aims to determine vehicle routes and schedules for the users who specify requests with pickup and delivery locations (Cordeau and Laporte, 2007) and has various applications, including freight transportation (e.g., Dumas et al., 1991), and elderly or disabled personnel transportation (e.g., Madsen et al., 1995; Melachrinoudis et al., 2007; Beaudry et al., 2010). There are two differences between the DARP and the ride-hailing sharing problem. First, dial-a-ride vehicles start from the depot(s) to pick up passengers in the DARP, while the start locations of ride-hailing vehicles can be anywhere in the ride-hailing sharing problem. Second, besides the time window constraint that the DARP focuses on, the ride-hailing sharing problem requires considering the travel cost (fare) constraint for each passenger to ensure that the out-of-pocket cost of each passenger is lower in a shared vehicle than a non-shared vehicle. It is noted that in the literature, the taxi sharing problem is a special type of the ride-hailing sharing problem, in which the ride-hailing sharing services may include not only taxis but also private cars (e.g., Didi).

Ho et al. (2018) pointed out that the DARP problem could be classified into four categories: static-deterministic, static-stochastic, dynamic-deterministic, and dynamic-stochastic. If the existing plans can (cannot) be modified when new information enters the system, the problem is dynamic (static). If the information received is certain (is unknown or uncertain) when making a decision, the problem is deterministic (stochastic). The ride-hailing sharing problem is a dynamic-deterministic problem. As ride-hailing customers who are willing to share a ride always want to match a ride-hailing vehicle as soon as possible and their requests can enter the system at random times, they often match drivers on very short notice. Previous studies have adopted several strategies to deal with the dynamic nature of the ride-hailing sharing problem. One strategy is that the model processes a request immediately after the system receives the request (Ma et al., 2013). Though the customers can get feedback in a short time, this strategy usually provides "shortsighted" solutions as it does not consider the influence of near-future requests, and thus leads to poor solution quality. Another strategy is to adopt the rolling horizon strategy, in which the solutions are determined using all known information within a planning horizon, but the final decisions have not been made until necessitated by a deadline of the requests (Agatz et al., 2011). This strategy can obtain a better solution than the first strategy as it considers more information, but the customers require longer time waiting for the final matching results. To balance the waiting time for matching results and solution quality, this study adopts the strategy that the dynamic problem is divided into small continuous static subproblems (Santos and Xavier, 2015; Alonso-Mora et al., 2017). Each static subproblem handles a scene corresponding to a specific time interval. This strategy can handle multiple requests simultaneously, and the time interval we set is short enough such that the customers do not wait too long for receiving feedback.

193
194 **Table 1 Characteristics for representative DARPs and ride-hailing sharing problems**

| Reference | Type | Objective(s) | Constraint(s)[2] | Scenario | Solution method(s) |
|---|---|---|---|---|---|
| Psaraftis (1980) | DARP | Minimize a weighted sum of the total travel time and dissatisfaction of customers | Capacity and MPS | D/S | Dynamic programming |
| Jaw et al. (1986) | DARP | Minimize a weighted sum of disutility to the system's customers and of operator costs | Time and capacity | S | Advanced dial-a-ride with time windows (ADARTW) heuristic |
| Madsen et al. (1995) | DARP | Multiple objectives[1] | Time and capacity | D | REBUS heuristic |
| Horn (2002) | DARP | Minimize total travel time while maximizing ridership using a weighted sum approach | Time and capacity | D | L2sched system |
| Attanasio et al. (2004) | DARP | Minimize total routing cost | Time and capacity | D | Tabu search |
| Coslovich et al. (2006) | DARP | Maximize the number of served customers | Time | D | Two-phase insertion technique |
| Beaudry et al. (2010) | DARP | Minimize a weighted sum of total travel time, total lateness, and total earliness | Time and capacity | D | Two-phase heuristic procedure |
| Schilde et al. (2014) | DARP | Minimize the sum of tardiness, earliness, and travel time violations | Time and capacity | D | Metaheuristic solution approaches based on dynamic variable neighborhood search |
| Ma et al. (2013, 2015) | RHSP | Minimize the total travel distance | Time, capacity, and cost | D | Dual-side vehicle searching algorithm |
| Santos and Xavier (2015) | RHSP | Maximize the number of served requests while minimizing the travel cost ratio | Time, capacity, and cost | D | GRASP with path relinking |
| Jung et al. (2016) | RHSP | Minimize total passenger travel times; maximize system profit | Time and capacity | D | Nearest vehicle dispatch algorithm/ Insertion heuristic/ Hybrid Simulated Annealing |
| Alonso-Mora et al. (2017) | RHSP | Minimize the travel delay of all passengers while maximizing the number of served requests | Time and capacity | D | Greedy assignment with Mosek and parallel computing |
| Sayarshad and Gao (2018) | DARP | Maximize social welfare | Capacity | D | A novel dynamic optimization algorithm with a Markov decision process |
| Wang et al. (2018) | RHSP | Minimize travel time | Time, capacity, and cost | D | A greedy strategy |
| Liang et al. (2020) | DARP | Maximize revenue, the number of matched customers while minimizing the fuel cost and delay | Time and capacity | D | A customized Lagrangian relaxation algorithm |
| This paper | RHSP | Maximize the weighted number of served customers while minimizing the weighted sum of the travel cost ratio and the travel time ratio | Time, capacity, and cost | D | Rolling horizon approach with MABC and path relinking |

195 Note: RHSP = Ride-hailing sharing problem, MPS = maximum position shift (i.e., the maximum difference
196 between the position of a customer in the sequence of deliveries/pickups and the first-come-first-served position
197 of that customer in the initial list of requests); D = dynamic problem, S = static problem, D/S = dynamic problem
198 and static problem; 1: The objective is formed by the mixture of objectives choosing from minimizing total driving

time, minimizing the number of vehicles, minimizing total waiting time, minimizing the deviation from promise service, and minimizing the total cost of operation of the vehicles. 2: time constraints refer to time window constraints, and cost constraints refer to travel cost constraints.

Table 1 summarizes the characteristics of the existing DARPs and ride-hailing sharing problems in the literature in terms of the problem type, design objectives, design constraints, operational scenarios, and solution methods. It can be seen that ride-hailing sharing problems have been studied in recent years, while DARPs have a long history. Different from the conventional DARPs, the ride-hailing sharing problem includes the passengers' travel cost (i.e., fare or out-of-pocket) into constraints to control the expense of each passenger on the trip due to the detour caused by ride-hailing sharing (e.g., Ma et al., 2013, 2015; Santos and Xavier, 2015). Regarding the design objectives, the commonly adopted objectives include maximizing the number of served customers (e.g., Coslovich et al., 2006) and minimizing the travel time (distance or delay) (e.g., Attanasio et al., 2004; Ma et al., 2013, 2015; Wang et al., 2018), whereas some studies formulated their design problems with more than one design objective (e.g., Jaw et al., 1986; Horn, 2002; Beaudry et al., 2010; Schilde et al., 2014; Santos and Xavier, 2015; Jung et al., 2016; Alonso-Mora et al., 2017; Sayarshad and Gao, 2018). On the other hand, passengers' travel cost is an important measure when providing ride-hailing sharing services as it often influences whether travelers choose a ride-hailing sharing service or just a ride-hailing service. Santos and Xavier (2015) formulated it in the form of the travel cost ratio. However, using this ratio in the objective function does not prevent the travel time increment due to the detour caused by ride-hailing sharing to be acceptable by passengers. Therefore, the objective function in our studied problem includes not only the travel cost ratio but also the travel time ratio that compares the actual travel time with the maximum allowable travel time (i.e., the maximum time that a passenger can spend for a ride) to limit the increase in travel time.

A wide range of solution methods have been proposed to solve the DARP and the ride-hailing sharing problem in the literature. Psaraftis (1980) developed an exact optimization procedure based on dynamic programming to solve the DARP (with ridesharing). Unlike the static version of the problem that does not consider the immediate requests, the dynamic version considers the immediate requests during the operation while it is limited to only the case with a single vehicle and many customers. The computational time of this algorithm is an exponential function of the number of customers. Alonso-Mora et al. (2017) built a request-trip-vehicle graph, which consisted of all possible combinations of the requests and vehicles according to the time window constraints. An integer linear program was formulated to determine the optimal assignment with the best objective function value (or the best objective value) based on the request-trip-vehicle graph. In the worst case, the method can be seen as an exhaustive search, so the parallel computations are used to speed up the method. However, the computational time of this method increases rapidly with the maximum waiting time. As DARPs and ride-hailing sharing problems are NP-hard (Baugh Jr., 1998; Santos and Xavier, 2015), exact methods are usually impossible to solve for optimal solutions in large instances efficiently. Heuristics or metaheuristics can search for near-optimal solutions efficiently and thus become widely adopted in the existing literature (e.g., Horn, 2002; Attanasio et al., 2004; Beaudry et al., 2010; Santos and Xavier, 2015; Jung et al., 2016). Meanwhile, many other methods are proposed to solve the DARP and ride-hailing sharing problem. For example, Ma et al. (2013) proposed a vehicle searching algorithm using a spatial-temporal index to find candidate vehicles and then a scheduling algorithm was applied to achieve matching and check constraints. This method for solving ride-hailing sharing problems is demonstrated to be very efficient and can be used in large-scale ride-hailing sharing problems. However, this method only suits the problem that minimizes the total travel distance or total travel time. New solution

methods may be required to handle other or more objectives. Sayarshad and Gao (2018) divided the DARP problem into multiple traveling salesman problems and solved them by a traveling salesman problem with pickup and deliver (TSPPD) algorithm. The Markov decision process was used to obtain information to calculate social welfare. Liang et al. (2020) solved the problem using a customized Lagrangian relaxation algorithm, and this algorithm was time-consuming (15.9 min for 50 iterations for a network with 66 road links and 46 nodes), leading to long waiting time for customers. Please refer to more comprehensive reviews on the solution methods of DARPs by Berbeglia et al. (2010) and Ho et al. (2018).

As reviewed by Ho et al. (2018), recently proposed metaheuristics have not been adopted in solving the DARP and its variants. As one of the recent methods mentioned in their review, the artificial bee colony (ABC) algorithm is adopted to solve our proposed problem. As a powerful metaheuristic proposed by Karaboga (2005), the ABC algorithm has been demonstrated with good performance in solving many problems, including numerical function optimization (e.g., Karaboga and Ozturk, 2009), structural inverse analysis (e.g., Karaboga, 2009), pattern classification (e.g., Karaboga and Ozturk, 2009), the leaf-constrained minimum spanning tree problem (e.g., Singh, 2009), and so on. It has also been applied in solving different logistics and transportation problems with satisfactory performance, such as capacitated vehicle routing problems (e.g., Szeto et al., 2011), return restriction design problems (e.g., Long et al., 2014), transit routes and frequency settings (e.g., Szeto and Jiang, 2014), and bicycle repositioning problems (e.g., Szeto and Shui, 2018). The works provide firm ground to apply the ABC algorithm in solving our dynamic ride-hailing sharing problem.

Unlike the literature, to solve the proposed dynamic ride-hailing sharing problem, we first decompose the whole planning horizon evenly into smaller time intervals and then adopt the ABC algorithm with path relinking in each time interval. Path relinking is an enhancement strategy proposed by Glover (1997) to explore the better solution between elite solutions obtained by tabu search or scatter search (e.g., Glover, 1997; Glover et al., 2000). Applying path relinking into the GRASP has significantly improved the solution time and quality (Resendel and Ribeiro, 2005). The GRASP with path relinking was first used in the ride-hailing sharing problem and achieved better performance than that without path relinking (Santos and Xavier, 2015). Furthermore, to speed up the matching between the requests and the vehicles, the vantage-point tree (VP tree) was used to do range queries to search for feasible vehicles around the origin of the request within a given radius. As a data structure for partitioning general metric space in a hierarchical way proposed by Yianilos (1993), the VP tree was widely used for efficient nearest neighbor queries (Nielsen et al., 2009; Fu et al., 2000). The resultant solution method is referred to as the MABC algorithm.

## 3. Dynamic Ride-hailing Sharing Problem

In this section, we first present the notations adopted in this problem and then give a detailed problem statement of the dynamic ride-hailing sharing problem. Afterward, the mathematical formulation of the static subproblem is presented in detail.

### 3.1. Notations

The notations used in this paper are listed as follows.

***Sets/indices***

| | |
|---|---|
| $V$ | Set of all vertices (points) on the road network; |
| $E$ | Set of all edges on the road network; |

| | |
|---|---|
| $\mathbb{N}$ | Set of all requests that are waiting to match; |
| $\mathbb{Q}$ | Set of all matched requests; |
| $\mathbb{Z}$ | Set of all ride-hailing vehicles available in the system; |
| $W$ | Set that contains all origins and destinations of requests in $\mathbb{N}$; |
| $U$ | Set that contains all origins and destinations of requests in $\mathbb{Q}$; |
| $i$ | Request $i$; |
| $i^+$ | Origin of request $i$, $i^+ \in V$; |
| $i^-$ | Destination of request $i$, $i^- \in V$; |
| $j$ | Ride-hailing vehicle $j$; |
| $j^+$ | Starting point of vehicle $j$, $j^+ \in V$; |
| $j^-$ | Dummy destination of vehicle $j$; |
| $R_j$ | Route of vehicle $j$, where $R_j = \left\{v_0^j, v_1^j, \ldots, v_{Z_j}^j\right\}$; |
| $v_z^j$ | The $z$ th point in the route of vehicle $j$, $v_z^j \in V$; |
| $i(v_z^j)$ | Request $i$ with either pickup or delivery at point $v_z^j$. |

***Parameters***

| | |
|---|---|
| $p_i$ | Number of passengers of request $i$; |
| $T_i^{\text{order}}$ | Order time of request $i$ (i.e., time of customers making request $i$); |
| $T_i^{\text{p}}$ | Latest pickup time of request $i$; |
| $T_i^{\text{d}}$ | Latest delivery time of request $i$; |
| $cost_i^{\text{dir}}$ | Total passengers' out-of-pocket cost of request $i$ through the direct trip without sharing ride-hailing vehicles; |
| $Z_j$ | Total number of pickup and delivery points in the route of vehicle $j$; |
| $q_j$ | Capacity of ride-hailing vehicle $j$; |
| $s_{v_z^j}$ | Service time at point $v_z^j$; |
| $t_{v_z^j, v_{z+1}^j}$ | Shortest travel time from $v_z^j$ to $v_{z+1}^j$ in the route of vehicle $j$; |
| $c_{v_z^j, v_{z+1}^j}$ | Fare from $v_z^j$ to $v_{z+1}^j$ in the route of vehicle $j$; |
| $b_1$ | Weight for each request; |
| $b_2$ | Weight for the travel cost ratio; |
| $b_3$ | Weight for the travel time ratio; |
| $DT_i$ | Shortest travel time from the origin to the destination of request $i$ without sharing ride-hailing vehicles; |
| $u_{now}^j$ | Location of vehicle $j$ at the beginning of the current time interval; |
| $BT_u$ | Arrival time at $u \in U$ before any new requests were inserted into routes; |
| $j(u)$ | Matched vehicle for $u \in U$ before any new requests were inserted into routes. |

***Decision Variables***

| | |
|---|---|
| $AT_{v_z^j}$ | Arrival time of vehicle $j$ at $v_z^j$; |
| $P_{v_z^j}$ | Number of passengers on vehicle *j after leaving* point $v_z^j$; |

$X_{u,v}^{j}$ 1 if the route of vehicle $j$ passes through vertex $v$ immediately after vertex $u$; 0, otherwise;

***Variables***

$cost_i^{\text{real}}$ Actual total passengers' out-of-pocket cost associated with request $i$;

$TT_i$ Total waiting and in-vehicle travel times associated with request $i$;

$pc_{v_z^j, v_{z+1}^j}^{i}$ Cost of the passengers of request $i$ from $v_z^j$ to $v_{z+1}^j$.

## 3.2. Problem statement

### *3.2.1. Inputs*

We consider a ride-hailing service provided by a private company. Let $G(V, E)$ be a complete undirected graph representing the road network. The ride-hailing sharing problem starts with a set of requests waiting to match and a set of ride-hailing vehicles currently available on the road network. Each request contains the information related to the origin, destination, order time, and number of passengers. The order time of request $i$ is associated with the latest pickup time $T_i^{\text{p}}$ and the latest delivery time $T_i^{\text{d}}$. The matched vehicle should pick up the passengers of request $i$ at point $i^+$ no later than $T_i^{\text{p}}$ and drop them off at point $i^-$ no later than $T_i^{\text{d}}$. According to $i^+$ and $i^-$, the shortest distance can be determined and then converted to the cost of the direct trip $cost_i^{\text{dir}}$ by multiplying the distance by ride-hailing fare per distance. This cost of the direct trip is also the upper bound of the ride-hailing sharing trip of request $i$ (after considering cost sharing of all passengers) to ensure that each passenger would not pay more by ride-hailing sharing than by making a direct trip. Each vehicle has its own information, including the starting point, capacity, and occupancy status.

### *3.2.2. Routes of vehicles*

The route of vehicle $j$ consists of starting point $v_0^j$ and other points (from $v_1^j$ to $v_{Z_j}^j$) corresponding to the origins or destinations of the requests that are served by vehicle $j$. The points in the route are arranged in chronological order, and the destination of a request is definitely after its corresponding origin in the route. Except for the starting point, other points in the route are associated with the information of the arrival time, load, and corresponding request (whose origin or destination is located at this point) in order to check the time window and capacity constraints during ride-hailing sharing. The arrival time of the vehicle at point $v_z^j$, denoted as $AT_{v_z^j}$, can be determined only when the arrival time of the vehicle at the previous point, the travel time between those two points $t_{v_{z-1}^j, v_z^j}$, and service time $s_{v_{z-1}^j}$ are known, which is expressed as $AT_{v_{z-1}^j} + t_{v_{z-1}^j, v_z^j} + s_{v_{z-1}^j}$. $AT_{v_z^j}$ has to obey corresponding pickup and delivery time windows, which implies that $AT_{v_z^j}$ must be less than $T_{i(v_z^j)}^{\text{p}}$ if $v_z^j$ is the origin of request $i(v_z^j)$, and must be less than $T_{i(v_z^j)}^{\text{d}}$ if $v_z^j$ is the destination of request $i(v_z^j)$. For the load $P_{v_z^j}$ after leaving point $v_z^j$, it is expressed as $P_{v_{z-1}^j} + p_{i(v_z^j)}$ and $P_{v_{z-1}^j} - p_{i(v_z^j)}$ if $v_z^j$ is the origin and

the destination of request $i(v_z^j)$, respectively. $P_{v_z^j}^j$ must not exceed vehicle capacity $q_j$.

### *3.2.3. Cost allocation*

A critical part of ride-hailing sharing is the calculation of the sharing cost for each request. Due to the complexity of the ride-hailing sharing pattern, it is impossible to determine the exact cost of serving passengers of a request in advance until all passengers in this request finish their trips. In this study, we adopt the equal-cost division principle on each pair of adjacent points as the cost allocation strategy. For each pair of adjacent points, the fare between those two points is allocated equally by all passengers traveling on this route segment. If request $i$ is served by vehicle $j$ between $v_z^j$ and $v_{z+1}^j$, we can obtain $pc_{v_z^j,v_{z+1}^j}^i = \frac{c_{v_z^j,v_{z+1}^j} \times p_i}{P_{v_z^j}^j}$. The actual total cost $cost_i^{\text{real}}$ spent by the passengers of request $i$ in their whole trip is the sum of the cost spent in all route segments that they pass through, which means $cost_i^{\text{real}} = \sum_z pc_{v_z^j,v_{z+1}^j}^i$, where the range of $z$ is determined by the route segments of vehicle $j$ that the passengers of request $i$ travel through. As the main reason for customers choosing ride-hailing sharing services is to reduce their out-of-pocket cost, the total cost $cost_i^{\text{real}}$ spent on the trip must be equal to or less than the cost of the direct trip $cost_i^{\text{dir}}$ without sharing.

### *3.2.4. Dynamic problem setting*

Not all requests are received at the beginning of the modeling horizon, and we cannot know the timing of receiving new requests in advance as in practice. Therefore, we cannot solve the dynamic ride-hailing sharing problem as a whole. Instead, we divide the modeling horizon into many intervals of equal length and divide the problem into many consecutive static ride-hailing sharing subproblems. Each subproblem corresponds to one time interval. The subproblems are solved in chronological order.

Define the current time interval as the interval associated with the subproblem concerned or to be solved. Before this interval, some requests were received. Some of them have not been served and are still waiting to be served. During the current time interval, all requests waiting to be served are handled simultaneously by solving the corresponding static ride-hailing sharing subproblem.

A longer time interval considers more requests at each execution, which leads to better matching performance, whereas the passengers require waiting longer to obtain the final matching result. Therefore, setting the time interval requires balancing both the matching performance and the users' waiting time.

In this paper, for simplicity, we assume that each vehicle stays in the last drop-off location to wait for requests assigned by the system if there are no passengers to deliver or pick up. This assumption can be easily relaxed by adding an endpoint to the vehicle route. We also assume that all customers are willing to share vehicles with others. Moreover, we do not consider the effect of traffic signals and assume uniform speed, and thus the travel time and the fare between two vertices are in proportion to the travel distance and remain unchanged throughout the modeling horizon. At the beginning of each time interval, the new requests received in the last time interval are collected by the system and added into set $\mathbb{N}$. At the end of each time interval, the requests that have already been matched to vehicles or the order times of the requests that

exceed their corresponding latest pickup times are removed from set $\mathbb{N}$. Set $\mathbb{Z}$ contains all the vehicles available during the current time interval. The routes of all available vehicles are inherited from the matching results of the last time interval. When a vehicle is available for ride-hailing sharing services, the vehicle is added to set $\mathbb{Z}$. The vehicles are removed from $\mathbb{Z}$ if they are not available. The new requests can be added to any positions of the routes if no constraints are violated. Note that the requests that were matched in the previous time intervals cannot be removed from the routes because the notices of matching results had already been sent to the corresponding vehicles and customers. The dynamic ride-hailing sharing problem is formed by linking consecutive static ride-hailing sharing subproblems.

### 3.3. Mathematical model of the static ride-hailing sharing subproblem

The static subproblem starts with set $\mathbb{N}$ and set $\mathbb{Z}$. Set $\mathbb{N}$, which contains all requests waiting to be matched, consists of (a) the new requests whose order times are during the last time interval and (b) the previous unmatched requests that were received earlier than the last time interval and have no matched vehicles in the previous time intervals, while the current time does not exceed the latest pickup times of the requests. Set $\mathbb{Z}$ contains all available vehicles during the current time interval.

To formulate this problem, we introduce dummy destination $j^-$ for all vehicles. Let $W = \{i^+, i^- | i \in \mathbb{N}\}$ be a set that contains all origins and destinations of requests in $\mathbb{N}$. Set $\mathbb{Q}$ contains all matched requests. $U = \{i^+, i^- | i \in \mathbb{Q}\}$ is the set that contains all origins and destinations of requests in $\mathbb{Q}$. $u \in U$ is associated with two pieces of information: arrival time $BT_u$ and matched vehicle $j(u)$.

To formulate the subproblem, we define the current location $u_{now}^j \in V$ based on the current time. The current time $T_c$ is set as the time at the end of the time interval in concern, not the beginning of the time interval. Then, the current location $u_{now}^j \in V$ is set as the first vertex that vehicle $j$ will visit after $T_c$ on the road network. The reason for those settings is that the routes in the latter time intervals, not the current time interval, can be adjusted. As shown in Figure 1, the large circles represent the origins or destinations of requests, and the small circle represents a point in set $V$. Meanwhile, the diamond shape represents the starting point of the route. The current location $u_{now}^j$ is not necessarily the origin or destination of a request; it can be the point in set $V$ between two adjacent pickup (or delivery) points where the vehicle passes through.

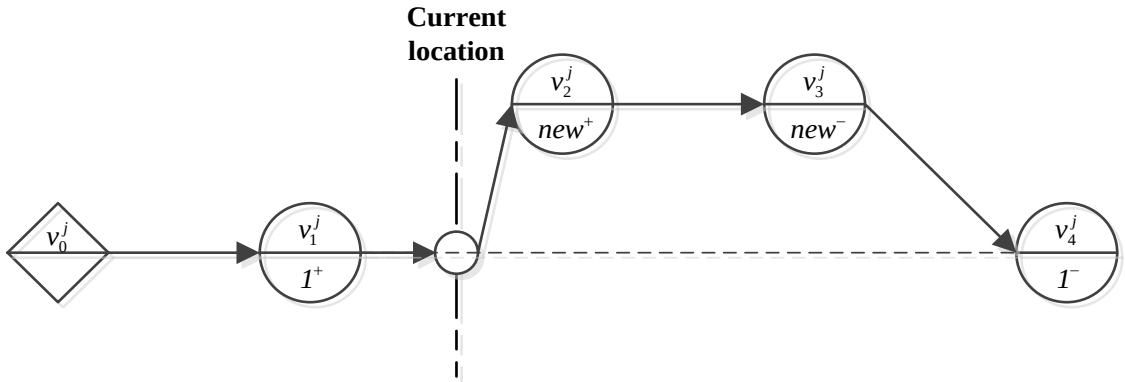



**FIGURE 1 An example of inserting a new request into a vehicle route**

The mathematical model of the static ride-hailing sharing subproblem in each time interval is shown as follows:

$$\max f\left(X_{u,v}^{j}, P_{v_z^j}, AT_{v_z^j}\right)=\sum_{i\in\mathbb{N}}\sum_{j\in\mathbb{Z}}\sum_{v\in W\cup U} X_{i^+,v}^{j}\left(b_1 p_i - b_2\frac{cost_i^{\text{real}}}{cost_i^{\text{dir}}} - b_3\frac{TT_i}{T_i^{\text{d}}-T_i^{\text{order}}}\right) \tag{1}$$

subject to

$$X_{u,v}^{j}\in\{0,1\},\ \ \forall j\in\mathbb{Z},\ \ \forall u,v\in W\cup U\cup\{j^+,j^-\}; \tag{2}$$

$$\sum_{j\in\mathbb{Z}}\sum_{v\in W\cup U} X_{i^+,v}^{j}\le 1,\ \ \forall i\in\mathbb{N}; \tag{3}$$

$$\sum_{v\in W\cup U\cup\{j^-\}} X_{j^+,v}^{j}=1,\ \ \forall j\in\mathbb{Z}; \tag{4}$$

$$\sum_{u\in W\cup U\cup\{j^+\}} X_{u,j^-}^{j}=1,\ \ \forall j\in\mathbb{Z}; \tag{5}$$

$$\sum_{v\in W\cup U\cup\{j^+\}} X_{v,u}^{j}-\sum_{v\in W\cup U\cup\{j^-\}} X_{u,v}^{j}=0,\ \ \forall j\in\mathbb{Z},\forall u\in W\cup U; \tag{6}$$

$$\sum_{v\in W\cup U} X_{i^+,v}^{j}-\sum_{v\in W\cup U} X_{v,i^-}^{j}=0,\ \ \forall j\in\mathbb{Z},\forall i\in\mathbb{N}\cup\mathbb{Q}; \tag{7}$$

$$\sum_{v\in W\cup U\cup\{j^-\}} X_{u,v}^{j(u)}=1,\ \ \forall u\in U; \tag{8}$$

$$(BT_u\le BT_{u_{now}^{j(u)}})\Rightarrow AT_u=BT_u,\ \ \forall u\in U; \tag{9}$$

$$(BT_v>BT_{u_{now}^{j(v)}})\wedge(X_{u,v}^{j}=1)\Rightarrow AT_v=AT_u+t_{u,v}+s_u,$$
$$\forall j\in\mathbb{Z},\forall u\in W\cup U\cup\{j^+,j^-\},\forall v\in U; \tag{10}$$

$$(BT_u\le BT_v)\wedge(j(u)=j(v))\Rightarrow AT_u\le AT_v,\ \ \forall u,v\in U; \tag{11}$$

$$(X_{u,v}^{j}=1)\Rightarrow AT_v=AT_u+t_{u,v}+s_u,$$
$$\forall j\in\mathbb{Z},\forall u\in W\cup U\cup\{j^+,j^-\},\forall v\in W; \tag{12}$$

$$(X_{i^+,v}^{j}=1)\Rightarrow BT_{u_{now}^{j}}\le AT_{i^-}\le AT_{i^+},\ \ \forall j\in\mathbb{Z},i\in\mathbb{N}; \tag{13}$$

$$0\le AT_{i^+}\le T_i^{\text{p}},\ \ \forall i\in\mathbb{N}; \tag{14}$$

$$0\le AT_{i^-}\le T_i^{\text{d}},\ \ \forall i\in\mathbb{N}; \tag{15}$$

$$(X_{u,v}^{j}=1)\wedge(v=i^{+})\Rightarrow P_{v}^{j}=P_{u}^{j}+p_{i},$$

$$\forall j\in\mathbb{Z},\forall i\in\mathbb{N}\cup\mathbb{Q},\forall u,v\in W\cup U\cup\{j^{+},j^{-}\};\tag{16}$$

$$(X_{u,v}^{j}=1)\wedge(v=i^{-})\Rightarrow P_{v}^{j}=P_{u}^{j}-p_{i},$$

$$\forall j\in\mathbb{Z},\forall i\in\mathbb{N}\cup\mathbb{Q},\forall u,v\in W\cup U\cup\{j^{+},j^{-}\};\tag{17}$$

$$P_{u}^{j}\le q_{j},\quad\forall j\in\mathbb{Z},\forall u\in W\cup U;\tag{18}$$

$$(X_{u,v}^{j}=1)\wedge(AT_{i^{+}}\le AT_{u}<AT_{i^{-}})\Rightarrow pc_{u,v}^{i}=\frac{c_{u,v}\times p_{i}}{P_{u}^{j}},$$

$$\forall j\in\mathbb{Z},\forall i\in\mathbb{N}\cup\mathbb{Q},\forall u,v\in W\cup U;\tag{19}$$

$$cost_{i}^{\text{real}}=\sum_{u,v\in V}pc_{u,v}^{i}\le cost_{i}^{\text{dir}},\quad\forall i\in\mathbb{N};\tag{20}$$

$$TT_{i}=AT_{i^{-}}-T_{i}^{\text{order}},\quad\forall i\in\mathbb{N}.\tag{21}$$

Objective function (1) consists of three terms: the number of served customers, travel cost ratio, and travel time ratio. $b_1$, $b_2$, and $b_3$ are all positive weight coefficients that define the relative importance of these three components, respectively. It is noted that the smaller $cost_i^{\text{real}}/cost_i^{\text{dir}}$ is, the more cost savings are; it is also noted that the smaller $\frac{TT_i}{T_i^{\text{d}}-T_i^{\text{order}}}$ is, the smaller increment in travel time compared with no ride-hailing sharing case is. Therefore, the objective value is larger if we get more matched requests, more savings in costs, and a smaller increment in travel time. Moreover, the objective value is larger when the matched request is associated with more passengers.

Constraint (2) defines $X_{u,v}^{j}$ to be binary. Constraint (3) guarantees that a request can only be matched by at most one vehicle. Constraints (4) and (5) ensure that each vehicle has an origin and a destination in its route. Constraint (6) is a flow conservation constraint to make sure that $u\in W\cup U$ served by a vehicle must have one point on the route before and after $u$. Constraint (7) ensures that the origin and destination must be served by the same vehicle if the request is served. Constraint (8) ensures that the matching between requests and vehicles formed by previous time intervals cannot be changed. Constraints (9)-(11) guarantee that the order of the visited points in the route inherited from the last time interval is not changed after inserting new requests. Constraints (12) and (13) ensure that new requests can only be inserted after the current locations of the vehicles. Constraints (14) and (15) are the time window constraints for all requests, while constraints (16)-(18) are the capacity constraints. Constraint (19) calculates the passengers' travel cost of each route segment, whereas constraint (20) ensures that the total passengers' travel cost of each request is not higher than the total passengers' travel cost without ride-hailing sharing. Constraint (21) calculates the total travel time of passengers of request $i$ if they are matched in this interval.

In our proposed model, the origins and destinations of new requests are allowed to be inserted anywhere in the vehicle route after the current vehicle location if the time, cost, capacity constraints are satisfied. However, the model proposed by Santos and Xavier (2015) allowed new requests to be added only after the destination point of the last delivered passenger aboard if there are passengers aboard the vehicle at the current time. Moreover, constraint (8) ensures that the matching between requests and vehicles formed by previous time intervals cannot change so that passengers just need to accept the matching results once. This constraint cannot

be found in their model. Furthermore, both travel time and cost ratios are considered in our objective function while Santos and Xavier (2015) only considered the travel cost ratio in their objective function. To sum up, our model is different from the counterparts of the related studies in terms of the objective function and constraints.

## 4. Solution Method

Like Santos and Xavier (2015) and Alonso-Mora et al. (2017), we solve the dynamic ride-hailing sharing problem by solving its subproblems in chronological order. Unlike Santos and Xavier (2015) and Alonso-Mora et al. (2017), we develop a method based on the modified artificial bee colony algorithm with path relinking to solve the static ride-hailing sharing subproblem for the time interval concerned.

### 4.1. Modified artificial bee colony algorithm with path relinking

#### *4.1.1. Basic artificial bee colony algorithm*

The ABC algorithm is an optimization algorithm that simulates the behavior of a honey bee swarm in search of food. The artificial bee colony consists of three groups of bees: employed bee, onlookers, and scouts, with the objective of finding the good food source(s). Each employed bee is responsible for one food source. It searches for food around a food source. The employed bees share the information on their best food sources found so far with the onlooker bees. Each onlooker then chooses a food source among those found by the employed bees by probability, where a more profitable (better) food source has a higher probability of being chosen. When the employed bee cannot find a better food source near the current source after some time, the employed bee turns to be a scout to exploit a new food source in the vicinity of the hive.

The ABC algorithm is a population-based heuristic, in which a food source represents a solution for the optimization problem, and the nectar amount of the food source represents the fitness of the corresponding solution. The ABC algorithm begins by generating a set of solutions randomly as the initial food sources, and each food source is assigned to an employed bee. After initial solutions are generated, employed bees, onlookers, and scouts exploit the food sources near the hive repeatedly during each iteration. In each iteration, each employed bee finds a new food source near the current source using a neighborhood operator and the nectar amount of the new food source (solution fitness, which is the increment on the objective function value in our study) is evaluated. If the nectar amount of the new food source is more than the old one, the employed bee abandons the current food source and is allocated to the new food source. Otherwise, the employed bee remains assigned to the current food source. Then each onlooker chooses a food source based on the nectar amount of the food sources (i.e., solution fitness) shared by employed bees by the roulette wheel selection method. Onlookers also exploit new food sources near the selected food sources using a neighborhood operator and evaluate the nectar amount of the new food sources. After all onlookers finish the exploitation process, the best new food source found by the onlookers near each food source of the employed bee is determined. If the nectar amount of the best new food source is more than the old one of the employed bee, the employed bee abandons the old food source and is assigned to the best new one. After that, if the nectar amount of a food source has not been improved for *limit* successive iterations, the employed bee becomes a scout, exploits a new food source randomly, becomes an employed bee again, and replaces the old food source with the new one. After all current food sources are checked, the new iteration of the ABC algorithm starts. The whole process is repeated to search for good solutions until the stop condition is reached. The steps of the ABC algorithm are presented as follows:

1. **Inputs:** Population size (number of food sources) $n$, maximum number of iterations $M$, and *limit* $L$.
2. Randomly generate a set of solutions as the initial food sources $x_i, i = 1,\ldots,n$. Each food source is allocated to an employed bee.
3. Calculate the fitness $f(x_i)$ of food source $x_i, i = 1,\ldots,n$. Set $l_i = 0, i = 1,\ldots,n$.
4. **For** iteration $m = 1$ to $M$, **do**

   **For** $i = 1$ to $n$, **do**

   i. Perform a neighborhood operator on the food source $x_i$ to determine a new food source $\tilde{x}_i$ near the food source.

   ii. If $f(\tilde{x}_i) > f(x_i)$, then replace $x_i$ with $\tilde{x}_i$ for the corresponding employed bee and $l_i = 0$, else $l_i = l_i + 1$.

   **End for**

   Set $G_i = \phi$, where $G_i$ is the set of new neighbor food sources of $x_i$ found by the onlookers.

   **For** $j = 1$ to $n$, **do**

   i. Select a food source $x_i$ using the roulette wheel selection method based on the fitness of all food sources.

   ii. Perform a neighborhood operator on $x_i$ to obtain a new food source $\tilde{x}_i$ near the food source.

   iii. $G_i = G_i \cup \tilde{x}_i$

   **End for**

   **For** $i = 1$ to $n$, **do**

   i. Select $\hat{x}_i = \arg\max_{x \in G_i} f(x)$, where $\hat{x}_i$ is the best food source in $G_i$.

   ii. If $f(\hat{x}_i) > f(x_i)$, then replace $x_i$ with $\hat{x}_i$ for the corresponding employed bee and $l_i = 0$, else $l_i = l_i + 1$.

   **End for**

   **For** $i = 1$ to $n$, **do**

   i. If $l_i = L$, randomly determine a new food source $\tilde{x}_i$ and replace $x_i$ with $\tilde{x}_i$.

   **End for**

**End for**

### *4.1.2. Modified artificial bee colony algorithm*

The ABC algorithm is the main algorithm to determine routes of ride-hailing vehicles. To speed up finding a good solution, path relinking is embedded into the ABC algorithm to explore new and better solutions between two known solutions. It is implemented when a new current best solution is found while executing the ABC algorithm. To further improve the solution search efficiency, the vantage-point tree is deployed to reduce the solution search space and constructed before establishing the initial solution. Contraction hierarchies is incorporated into

the ABC algorithm to determine the shortest path in the studied problem. The detailed information about the vantage-point tree and path relinking will be described in Section 4.2 and Section 4.8, respectively. Figure 2 shows the procedure of the resultant solution method, namely the modified artificial bee colony algorithm. The algorithm ends when running time $T_{\text{run}}$ reaches the time limit, which is equal to the length of time interval $T_{\text{interval}}$.

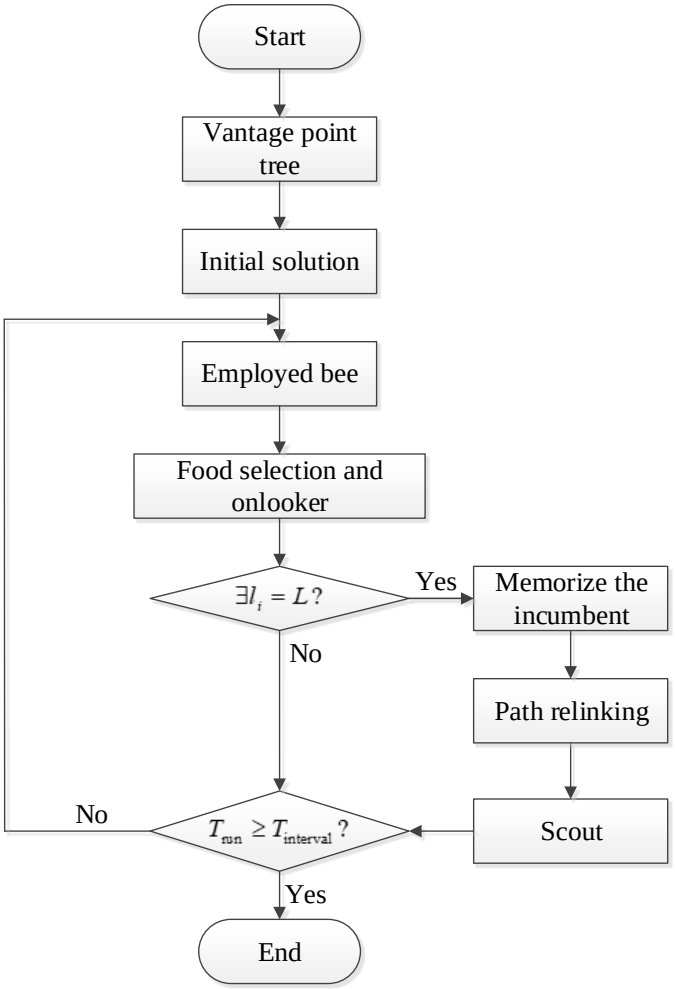



**FIGURE 2 Flowchart of the modified artificial bee colony algorithm**

## 4.2. Vantage-point tree

A VP tree is a metric tree that segregates the whole set of vertices of the network into small sets by choosing vantage points (i.e., vantage vertices). The VP tree is efficient in conducting the nearest neighbor search because the vertices are stored in the tree structure, and the search can happen only in small parts of the tree (Yianilos, 1993). The VP tree contains two important segments, including the construction segment and the searching segment. For the construction segment, a vantage point (*vp*) is determined to divide all the vertices into two smaller parts during each partition. The vertices whose distance to the vantage point is less than a threshold (*mu*) are stored in the *left* sub-tree, and the vertices whose distance to the vantage point is larger than the threshold are stored in the *right* sub-tree. Each node in the tree stores the information of the vantage point and threshold. A tree data structure is created by recursively implementing this procedure to divide the data starting from the root. After building the tree using the construction segment, the searching segment is executed at the beginning of the static

subproblem, when the information of new requests is known. In this study, the range nearest neighbor search is used in the searching segment, in which we want to determine all the vertices within the radius *tau* around the query vertex $q$. Let the distance between any two vertices *m* and *n* be $\mathrm{dist}(m,n)$. The pseudo-code of the VP tree in this paper is summarized as follows:

**Part 1: Construction segment**

1. **Inputs:** A set *S* containing all vertices.
2. Build_VP_tree ( *S* ).
3. **Return** a VP tree.

**function** Build_VP_tree ( *S* ):

i. If $S = \varnothing$, then **return** $\varnothing$.
ii. Establish a new *node*: new(*node*).
iii. Determine the vantage point at the *node* (*node.vp* := Select_vp (*S*)) and determine the threshold at the *node* (*node.mu* := Median $(S, vp)$).
iv. Determine the set of vertices in the *left* sub-tree ( $L := \{s \in S - \{vp\} \mid \mathrm{dist}(vp, s) < mu\}$ ) and determine the set of vertices in the *right* sub-tree ( $R := \{s \in S - \{vp\} \mid \mathrm{dist}(vp, s) \geq mu\}$ );
v. Construct the new *node*s in the next level ( *node.left* := Build_VP_tree $(L)$, *node.right* := Build_VP_tree $(R)$ ).
vi. **return** *node*.

**function** Select_vp ( *S* ):

i. Choose a random sample $P$ from $S$.
ii. Set *best_spread* := 0.
iii. **For** $p \in P$, **do**
   1) Choose a random sample $D$ from $S$.
   2) Determine the value of *spread* ( *spread* := SecondMoment $(D, p)$ ).
   3) If *spread* > *best_spread*, then replace *best_spread* with *spread* and replace *best_p* with *p.*

   **End for**
iv. **return** *best_p*.

**function** Median $(S, p)$:

i. Sort $S$ in accordance with the distance from $p$.
ii. Determine threshold *mu* (the distance that is equal to the median among all distances from all vertices in *S* to *p*).
iii. **Return** *mu*.

**function** SecondMoment $(D, p)$:

i. Calculate $\mu$ with $\frac{\sum_{d \in D} \mathrm{dist}(d, p)}{N}$, where *N* is the number of vertices in *D*.
ii. Calculate *spread* with $\frac{\sum_{d \in D} (\mathrm{dist}(d, p) - \mu)^2}{N}$.
iii. **return** *spread*.

**Part 2: Searching segment**

1. **Inputs:** A query vertex $q$, the desired radius *tau*, and the VP tree.

2. Search_VP_tree (*root_node*).
3. **Return** the vertices around the query vertex within $tau$.

**procedure** Search_VP_tree (node)

i. If $node = \varnothing$, then, **return** $\varnothing$.

ii. If $\text{dist}(q, node.vp) - tau < mu$, then Search_VP_tree (*node.left*); if $\text{dist}(q, node.vp) + tau > mu$, then Search_VP_tree (*node.right*).

In this study, the VP tree is used to find the available vehicles around the origin of a request within a radius. The radius for each request is different because it is calculated based on the latest pickup and delivery times of the request, which means the vehicles outside this circle are impossible to pick up the customers of this request timely. At the beginning of each time interval, request $i$ has matching set $K_i$, the latest pickup time, and the last delivery time. Matching set $K_i$ of request $i$ is built by inserting all vehicles still available into the set. To determine the radius of request $i$, we define the maximum slack time of request $i$, which is the maximum time that a vehicle can spend to pick up customers from its current location to the origin of the request. The maximum slack time of request $i$ is equal to $\min(T_i^{\mathrm{p}} - T_c,\ T_i^{\mathrm{d}} - T_c - st_i^{o,d})$, where $T_i^{\mathrm{p}}$ is the latest pickup time, $T_i^{\mathrm{d}}$ is the latest delivery time, $T_c$ is the current time, and $st_i^{o,d}$ is the shortest travel time from the origin to the destination for request $i$. By multiplying the vehicle speed, the maximum slack time can be transformed into the maximum pickup distance, which is the radius of the circle with the origin of request $i$ as the center and can be used to distinguish infeasible matching between request $i$ and vehicles. All vehicles outside this circle at the current time are impossible to reach the origin or destination in time and hence they are removed from the matching set $K_i$. After that, request $i$ can only match the vehicles selected from $K_i$ during executing the algorithm to narrow the search space for each request and to improve the computational efficiency.

### 4.3. Solution representation

The solution (food source) of the MABC algorithm is a matching result between requests and drivers. The solution is formed by a set of routes of vehicles. Each route is represented in the form of a vector with the length of $1+2n$, in which $n$ is the number of requests matched to this vehicle route, and the first element in the vector means the starting point of this vehicle. Figure 3 illustrates a representation of a route for a ride-hailing vehicle after inserting a new request, in which point $0$ in the route is the starting point of the vehicle. Points $\mathrm{L}^+$ and $\mathrm{L}^-$ ($\mathrm{L}=1, 2, new$) represent the origin and destination of request $\mathrm{L}$, respectively. In MABC, each point in the route stores additional information, including the number of passengers aboard and the arrival time at this point.

| 0 | $1^+$ | $2^+$ | $1^-$ | $new^+$ | $new^-$ | $2^-$ |
|---|---|---|---|---|---|---|

**FIGURE 3 Solution representation of a vehicle route**

### 4.4. Initial solution

An empty set $M$ is created to store the matched new requests during the initial solution process. An initial solution is created by assigning a vehicle from the matching set $K_i$ randomly at a time to a request in $\mathbb{N}$ but not in $M$. The request is inserted into all possible

locations in the route of that vehicle. If there are insertions that satisfy both the time window and the capacity constraints, then the insertion with the maximum objective value is selected, the corresponding vehicle is assigned to this request, and this request is added into $M$. Otherwise, this request is skipped, and the next request in $\mathbb{N}$ but not in $M$ is chosen. Unlike the capacity and the time window constraints that are difficult to satisfy with the increasing detours caused by adding requests, travel cost constraints are usually satisfied—the travel cost of each request usually decreases due to ride-hailing sharing. Therefore, the travel cost constraints of the corresponding vehicle route for each request are checked only after the capacity and time window constraints are checked. If the travel cost constraints are not satisfied by the route of that vehicle, the requests in $\mathbb{N}$ that were assigned to this route are removed from $M$. This route is restored to the status without those new requests. The above procedure is repeated until $M$ collects all requests in $\mathbb{N}$, or a pre-defined maximum number of iterations is reached.

### 4.5. Selection of food sources

At each iteration of the MABC algorithm, each onlooker selects a food source based on the information shared by the employed bees. The method used in choosing a food source is the roulette-wheel selection method. The probability of choosing food source $x_i$ is equal to $p(x_i) = \frac{\Delta z(x_i)}{\sum_{i=1}^{n} \Delta z(x_i)}$, where $\Delta z(x_i)$ is the increment in the objective value of food source $x_i$ after inserting new requests. This increment means the contribution of the new requests to the objective value. A better solution has a larger increment.

### 4.6. Neighborhood operators

A neighborhood operator is applied to search for a new solution around the current solution. Two neighborhood operators, namely add operator and swap operator, are used in the MABC algorithm considering the characteristics of the ride-hailing sharing problem. Whenever employed bees or onlookers seek a new solution, one of the two neighborhood operators is used randomly. If a better solution is found by the neighborhood operator, the current solution is replaced with that new solution.

#### *4.6.1. Transfer operator*

The transfer operator randomly selects request $i$ from set $\mathbb{N}$. If request $i$ is matched a vehicle, the operator chooses vehicle $j$ different from the currently matched vehicle from the corresponding matching set $K_i$. Then an attempt to remove request $i$ from the original route and insert $i$ into the new route of vehicle $j$ is made (and the insertion method will be described in Section 4.7). If request $i$ has not been matched a vehicle, the request is randomly added to vehicle $j$ from the matching set $K_i$. In each case, if the insertion is feasible and the objective function of the new solution is larger than the current one, the new solution replaces the current one, and the *limit* count of this food source is reset as zero. Otherwise, the *limit* count increases by one.

#### *4.6.2. Swap operator*

Each swap operator randomly chooses two requests $i_1$ and $i_2$ from set $\mathbb{N}$ and their corresponding matched vehicles are $j_1$ and $j_2$, respectively. Before changing the vehicle routes, we require to check the matching sets of both requests. If $j_1$ is in the matching set

$K_{i_2}$ and $j_2$ is in the matching set $K_{i_1}$, $i_1$ and $i_2$ are removed from their original routes and then $i_1$ and $i_2$ are inserted into the new routes of vehicle $j_2$ and $j_1$, respectively (the insertion method will also be described in Section 4.7). If a new, feasible, and better solution is found by the swap operator, the current solution is replaced with the new solution, and the *limit* count of this new food source is set as zero; otherwise, the *limit* count increases by one. Note that if the condition that $j_1$ is in matching set $K_{i_2}$ and $j_2$ is in matching set $K_{i_1}$ is not satisfied, the transfer operator, instead of the swap operator, will be used for neighborhood search.

### 4.7. Insertion method

When request $i$ is matched to vehicle $j$, both origin $i^+$ and destination $i^-$ of request $i$ are required to insert into the route of $j$. The idea of inserting a request into the route is to check all possible insertion locations and then determine the best location to insert the request. There are two principles to determine the possible insertion locations: First, $i^+$ and $i^-$ can be inserted only after the points whose arrival time is later than the current time; second, origin $i^+$ must be located before destination $i^-$. For each possible insertion attempt, the vehicle route is restructured, and the distance between each pair of adjacent vertices (e.g., the origin of a request, the destination of a request, and the current location of the vehicle) in the route is calculated based on the shortest path between them using contraction hierarchies (Geisberger et al., 2008). Then we can recalculate the arrival time and the number of passengers right after the points of the vehicle route that are obtained after the insertion, as well as the travel cost of each request and the objective value. The points that have been influenced by the insertion are evaluated to check whether there are violations of the time window, capacity, and travel cost constraints. The new and feasible insertions are recorded, and the insertion that yields the best objective value among these insertions is chosen as the new solution to replace the old one. However, no insertion will be undergone if none of the possible insertions satisfied all the constraints.

Note that contraction hierarchies is a speed-up method for searching for the shortest path in a network. It is a two-phase approach consisting of the preprocessing and query phases. Contraction hierarchies has advantages of quick preprocessing times, low space requirements, and fast query times. Each query only needs to take a short time (microseconds). Therefore, it can be used in solving large-scale problems (Geisberger et al., 2012).

### 4.8. Path relinking

In this study, path relinking is incorporated into the ABC algorithm to improve solution quality. The idea of path relinking is to try to determine a new better solution between two known good solutions. Path relinking is performed when the *limit* count of at least one food source $s_2$ is equal to the limit $L$.

At the beginning of path relinking, there are two known good solutions, including incumbent $s_1$ and food source $s_2$ with the *limit* count equal to $L$. If $s_2$ is better than $s_1$, we replace $s_1$ with $s_2$ and stop. Otherwise, path relinking is applied to those two solutions. When executing path relinking, $s_1$ is set as the initial solution (i.e., $s = s_1$) and food source $s_2$ with the *limit* count equal to $L$ is set as the guiding solution. The initial solution is transformed into the guiding solution during path relinking by implementing a series of operations sequentially to

search for the new solutions (if any) that are better than both the initial and guiding solutions. The reason for choosing the best solution as the initial solution is that the new and better solutions are more probably found near incumbent $s_1$.

At each iteration of path relinking, the differences between $s$ and $s_2$ are first identified. There are three possible situations for each of the requests in $\mathbb{N}$, including: (a) the matched vehicles of the request in $s$ and $s_2$ are different, (b) the request matches a vehicle in $s$ but does not match any vehicles in $s_2$, and (c) the request does not match any vehicles in $s$ but matches a vehicle in $s_2$. The transformations for $s$ consist of (a) removing the request from the route of the matched vehicle in $s$ and inserting the request to the route of the matched vehicle in $s_2$ for the first situation, (b) removing the request from the vehicle route for the second situation, and (c) adding the request into the route of the matched vehicle in $s_2$ for the third situation. By revising each difference between $s$ and $s_2$ separately, we can get several new solutions compared to $s$. If all new solutions are not better than $s$, the path relinking procedure stops. Otherwise, the solution with the best improvement on the objective function value among new solutions is adopted and replaces $s$, and the next iteration begins. After path relinking, $s_1$ is replaced with $s$ found by path relinking.

## 5. Computational Experiments

This section presents the results and analyses involving the MABC algorithm for the ride-hailing sharing problem. The GRASP heuristic described by Santos and Xavier (2015) is taken as a benchmark, and its performance is used to compare with the performance of the MABC algorithm proposed in this paper. All experiments are performed on an Intel Core i7-4770 3.40 GHz CPU desktop computer, with 32 GB memory. The code is implemented in C++ with GCC (GNU Compiler Collection) using Linux (Ubuntu 16.04).

The ride-hailing fare is assumed to be proportional to travel distance and is set as one dollar per kilometer. The speed of all vehicles on all roads is set as 30 km/hour so that the travel time of each link can be calculated using the given link distance and speed. The coefficients of $b_1$, $b_2$, and $b_3$ are set as 2, 0.9, and 0.9, respectively. For simplicity, the service time at each pickup or delivery point is set as 0. Unless stated otherwise, the length of a time interval is set as 10 s. All experiments are executed 20 times, and the result of each experiment is obtained from the average value in 20 runs.

Each request may have several ride-hailing customers to take a vehicle in reality. However, we set the number of ride-hailing customers in each request equal to 1 for simplicity. We also set that the latest pickup time $T_i^{\mathrm{p}}$ is 5 min (i.e., 300 s) later than the order time $T_i^{\mathrm{order}}$, which means that ride-hailing customers cannot wait for more than 5 min at their origin. The latest delivery time $T_i^{\mathrm{d}}$ of request $i$ is set as the order time $T_i^{\mathrm{order}}$ plus the maximum allowable delay. To calculate the maximum allowable delay, we define $co_{delay}$ as the delay coefficient, which is greater than 1 and represents the tolerance of the passengers due to the increase in travel time caused by the detours in ride-hailing sharing. The maximum allowable delay is equal to the product of $co_{delay}$ and the shortest travel time between the origin and destination of request $i$ (which can be determined by the contraction hierarchies highlighted in Section

4). Unless specified otherwise, the delay coefficient $co_{delay}$ is 1.3.

### 5.1. Data description

The ride-hailing request data in Chengdu, China, are used in the computational experiments. Chengdu is a typical city with a circular layout, centered on Tianfu Square, which can also be reflected in the layout of the road network in Chengdu. The map of Chengdu is shown in Figure 4. The whole city is connected by a "ring and radial" highway network, and the circular road network divides the city into multiple regions. The Chengdu map data used in our study is downloaded from Open Street Map, which consists of 34,186 vertices and 78,157 edges. This map data are used to create a road network for the ride-hailing sharing problem.

The number of orders varies over time of day, and there are few orders in the night time. To illustrate a worst-case scenario, we process one-hour ride-hailing request data between 0:00 am and 1:00 am on November 1, 2016, in Chengdu, China, obtained from Didi. The total number of orders is 3,661 in one hour. Each order has a set of information, including the order ID, order time, and longitudes and latitudes of the origin and destination. A sample of the ride-hailing data is shown in Table 2. The order ID is desensitized by Didi to protect the privacy, and the order time is represented through the time stamp. The geolocations are given according to the GCJ-02 coordinate. As shown in Figure 5, the spatial distributions of the request origins and destinations have a similar pattern, in which the orders are denser when they are closer to the city center.

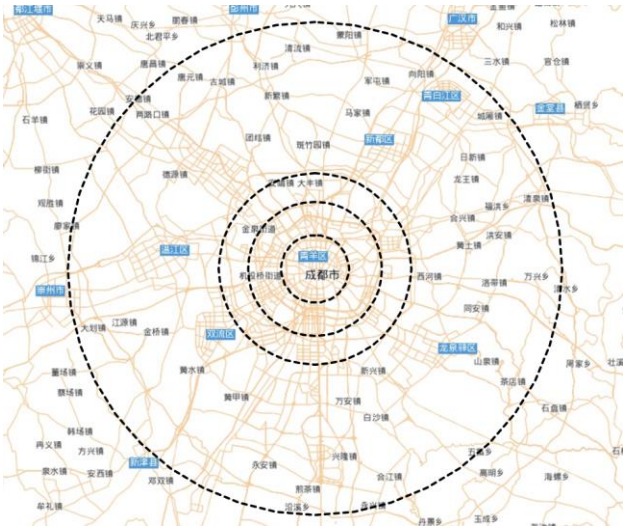



**Figure 4 The map of Chengdu, China**

**Table 2 A sample of the ride-hailing data in Chengdu, China**

| Order ID | Order time | Longitude (origin) | Latitude (origin) | Longitude (destination) | Latitude (destination) |
|---|---|---|---|---|---|
| fbcgi49b7j5yv | 1477964797 | 104.09464 | 30.703971 | 104.08927 | 30.65085 |
| 48adc4bhcb6t | 1477985585 | 104.076509 | 30.76743 | 104.0637 | 30.58951 |
| aci8afhg8k@ | 1478004952 | 104.019699 | 30.689007 | 104.105324 | 30.66395 |
| 6fhhfe952dar | 1477989840 | 104.03609 | 30.62269 | 104.04386 | 30.68232 |
| bd7ea2ld3b7z | 1477958005 | 104.115997 | 30.652313 | 104.104421 | 30.695113 |
| …… | …… | …… | …… | …… | …… |

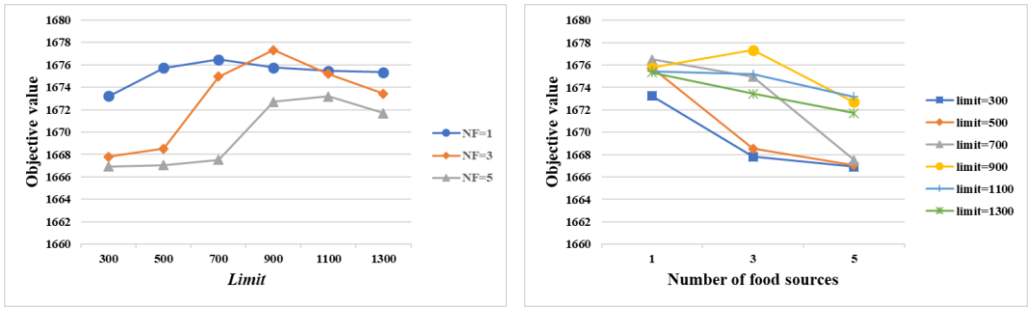
(a) Distribution of order origins (b) Distribution of order destinations

**Figure 5 Distributions of order origins and destinations in Chengdu, China**

The vehicles used in the study are randomly generated based on the regions, and the number of ride-hailing vehicles is similar to the real situation. We analyze the trajectory data of the ride-hailing vehicles in a day in Chengdu and determine the number of available vehicles in each hour of the day. The number of vehicles is around 2,400 between 0:00 to 1:00, so we generate 2,400 vehicles in this study to serve customers. Based on the circular road network, as shown in Figure 4, we can roughly divide Chengdu into five regions. Each ring road is approximately represented by a circle, and the center of all circles is the center of the city (Tianfu Square). The radii of the four concentric circles are approximately 4.8 km, 9.5 km, 13.6 km, and 35 km. The circles separate the city into five parts, in which region one is within the 4.8-km radius circle, region two is between the 4.8-km and the 9.5-km radius circles, region three is between the 9.5-km and the 13.6-km radius circles, region four is between the 13.6-km and the 35-km radius circles, and region five is beyond the 35-km radius circle. The distribution of vehicles in each region at the beginning is according to the number of request origins in this area. The ratios of request origins in each region are obtained to be 0.5865, 0.3189, 0.0744, 0.0299, and 0.0004, respectively. To determine the origin of each vehicle, a region is randomly selected using the above ratios as selection probabilities, and then the origin of each vehicle is randomly chosen from the vertices in the selected region.

### 5.2. Parameter tuning

The MABC has three parameters to tune, including the number of food sources, *limit*, and the maximum number of iterations. However, in the current dynamic ride-hailing sharing problem, the running time of the algorithm for each subproblem is equal to the time interval of each subproblem. Therefore, we set the stopping condition to be the length of the time interval instead of the maximum number of iterations. In other words, the algorithm must stop when the running time reaches the given length of the time interval, and the number of iterations can be ignored. Therefore, we require tuning only two parameters: the number of food sources and *limit*. Figure 6 displays the results of the parameter tuning with different numbers of food sources and *limit* values.




**(a)** **(b)**

**Figure 6 Parameter tuning with the time interval of 10 s and the delay coefficient of 1.3: (a) Tuning of *limit*; (b) Tuning of the number of food sources**

As shown in Figure 6(a), the objective value increases with *limit* until reaching a threshold. A too small *limit* value restricts the algorithm to obtain very good nearly local optima, whereas a too large *limit* value restricts the algorithm to explore more new solutions. As shown in Figure 6(b), the objective value decreases with the increasing number of food sources except for the limit of 900. By comparing the results, the best combination of the parameters is achieved when the number of food sources is 3, and the *limit* equals 900. Therefore, the subsequent sections adopt this setting. Providing that the number of requests is roughly uniform within an hour, when the time interval is longer, it is expected that the value of *limit* should be larger to handle the increase in the number of requests. Based on Figure 6, parameter *limit* is proportional to the length of time interval $e$, and this value is approximately equal to $90e$. Thus $limit = 90e$ is used in all the experiments in the following subsections.

### 5.3. Effect of the time interval

In ride-hailing sharing operations, customers are not willing to spend too much time on matching. However, the system requires time to collect information about the requests and vehicles and run the matching algorithm. In this study, we consider three time intervals, i.e., 10 s, 30 s, and 60 s. The maximum waiting time from placing a ride-hailing order by phone to receiving the matching result equals twice the length of the time interval. For instance, the maximum waiting time is 2 min for the time interval of 60 s, in which customers have to wait for 60 s for the operator of the ride-hailing service to collect all requests during the interval and another 60 s for waiting for the matching results.

To compare the results under different time interval lengths, seven performance measures are adopted, as shown in Table 3. The objective value is the most important and comprehensive measure, which evaluates the combined effects of the number of served customers, the travel cost ratio, and the travel time ratio simultaneously. The matching percentage is the percentage of the matched requests in the total requests collected. The sharing percentage is the percentage of requests involving ride-hailing sharing in all matched requests. The average out-of-pocket cost saving percentage per passenger ( $R_{money}$ ), due to the benefits of sharing ride-hailing fares, is expressed as

$$R_{money} = \frac{\frac{cost_{i'}^{dir} - cost_{i'}^{real}}{cost_{i'}^{dir}} \times p_{i'}}{\sum_{i' \in D} p_{i'}} \times 100\% = \frac{\sum_{i' \in D} \left( \frac{cost_{i'}^{dir} - cost_{i'}^{real}}{cost_{i'}^{dir}} \times p_{i'} \right)}{\sum_{i' \in D} p_{i'}} \times 100\%, \quad (22)$$

where $D$ is the set of matched requests and $cost_{i'}^{\text{real}}$ and $cost_{i'}^{\text{dir}}$ are the total passengers' out-of-pocket costs from the origin to the destination of request $i'$ with and without ride-hailing sharing, respectively. The percentage of total out-of-pocket cost saving ($RT_{money}$) is expressed as

$$RT_{money} = \frac{\sum_{i' \in D} \left( \left( \frac{cost_{i'}^{\text{dir}}}{p_{i'}} - \frac{cost_{i'}^{\text{real}}}{p_{i'}} \right) \times p_{i'} \right)}{\sum_{i' \in D} \left( \frac{cost_{i'}^{\text{dir}}}{p_{i'}} \times p_{i'} \right)} \times 100\% = \frac{\sum_{i' \in D} \left( cost_{i'}^{\text{dir}} - cost_{i'}^{\text{real}} \right)}{\sum_{i' \in D} cost_{i'}^{\text{dir}}} \times 100\% . \quad (23)$$

Since the travel fare per unit distance is fixed, $RT_{money}$ can also be interpreted as the percentage of total vehicle travel distance saving of the whole system. Regarding the increment in time, the average travel time increment percentage per passenger $R_{time}$, which is brought by the detours in ride-hailing sharing, is expressed as

$$R_{time} = \frac{\left( \sum_{i' \in D} \frac{t_{i'}^{\text{real}} - t_{i'}^{\text{dir}}}{t_{i'}^{\text{dir}}} \times p_{i'} \right)}{\sum_{i' \in D} p_{i'}} \times 100\% , \quad (24)$$

where $t_{i'}^{\text{real}}$ is the *actual* combined in-vehicle travel and waiting time and $t_{i'}^{\text{dir}}$ is the shortest travel time from the origin to the destination of request $i'$ without ride-hailing sharing. Moreover, the percentage of total travel time increment $RT_{time}$ is expressed as

$$RT_{time} = \frac{\sum_{i' \in M} \left( t_{i'}^{\text{real}} - t_{i'}^{\text{dir}} \right) \times p_{i'}}{\sum_{i' \in M} \left( t_{i'}^{\text{dir}} \times p_{i'} \right)} \times 100\% . \quad (25)$$

It is noted that $R_{money}$ and $R_{time}$ are respectively the out-of-pocket cost saving percentage and the travel time increment percentage based on individuals whereas $RT_{money}$ and $RT_{time}$ are the corresponding measures based on the whole system.

As shown in Table 3, a longer time interval results in a larger objective value. In this paper, we assume that all matched requests in the previous time intervals cannot be modified in later time intervals. Therefore, when the time interval is longer, the static subproblems have more chances to get better solutions. Meanwhile, a longer time interval can have a larger average out-of-pocket cost saving percentage per passenger, which means that ride-hailing sharing can generate more economic benefits (i.e., cost reduction) for each request when the length of the time interval is longer. Moreover, by comparing the values of $RT_{money}$ and $RT_{time}$, it can be seen that a longer time interval can result in a larger overall saving in money and a smaller overall increment in time. However, a longer time interval has no advantage in the matching percentage and sharing percentage, because a longer interval implies a lower frequency of matching, which leads to a higher probability of missing feasible vehicles for matching.

Regarding the general impact brought by ride-hailing sharing, Table 3 shows that the average out-of-pocket cost saving percentage per passenger $R_{money}$ can reach more than 26%, and the average travel time increment percentage per passenger $R_{time}$ is only around 15%. This means that passengers can use less proportion of extra travel time to exchange for a larger proportion of money-saving due to ride-hailing sharing. This implies that ride-hailing sharing is a good choice for those who have a high tolerance for time and want to save money.

Table 4 presents the t-test results of the differences in the average objective values between the experiments with different time intervals. From the results in Table 3 and Table 4, we can conclude that the differences in the objective values in different groups are significant, and the 60-second time interval achieves the best results, as it performs the best in most of the performance measures. These results imply that customers have to spend more time waiting for the matching results when the operator wants to improve system performance.

**Table 3 Performance comparison in terms of different time intervals**

| Time interval | Objective value | Matching | Sharing | $R_{money}$ | $R_{time}$ | $RT_{money}$ | $RT_{time}$ |
|---|---|---|---|---|---|---|---|
| 10 s | 1677.33 | **85.24%** | 72.10% | 26.32% | **15.48%** | 25.73% | 16.25% |
| 30 s | 1682.97 | 85.08% | **73.28%** | 26.70% | 15.56% | 25.73% | 16.23% |
| 60 s | **1689.94** | 85.15% | 73.11% | **26.87%** | 15.51% | **26.03%** | **16.14%** |

**Table 4 T-tests on the difference between average objective values**

| Test | Difference in mean | t-statistic | p-value |
|---|---|---|---|
| 10 s vs. 30 s | 5.64 | 5.34 | 0.00 |
| 30 s vs. 60 s | 6.97 | 5.04 | 0.00 |

### 5.4. Effect of path relinking

This section investigates the effect of the inclusion of path relinking into the resultant solution algorithm. Table 5 shows that all the differences in objective values with and without path relinking are statistically significant (as reflected from the p-value), and the inclusion of path relinking can significantly improve the solution quality regardless of the length of the time interval. Among all three time intervals, the solution algorithm with path relinking (i.e., the proposed MABC algorithm) can achieve a larger sharing percentage, a higher percentage of travel cost sharing per passenger, and a higher percentage of total out-of-pocket cost saving. Therefore, it is better to integrate path relinking into the solution algorithm to achieve a better solution.

**Table 5 Performance comparison of solution methods with or without path relinking**

| Time interval | PR | Objective Value | p-value | Matching | Sharing | $R_{money}$ | $R_{time}$ | $RT_{money}$ | $RT_{time}$ |
|---|---|---|---|---|---|---|---|---|---|
| 10 s | With PR | **1,677.33** | 0.00 | 85.24% | **72.10%** | **26.32%** | 15.48% | **25.73%** | 16.25% |
| | Without PR | 1,668.81 | | **85.27%** | 71.35% | 25.84% | **15.35%** | 25.11% | **16.01%** |
| 30 s | With PR | **1,682.97** | 0.00 | **85.08%** | **73.28%** | **26.70%** | 15.56% | **25.73%** | 16.23% |
| | Without PR | 1,675.86 | | 84.94% | 72.71% | 26.43% | **15.50%** | 25.47% | **16.15%** |
| 60 s | With PR | **1,689.94** | 0.04 | **85.15%** | **73.11%** | **26.87%** | **15.51%** | **26.03%** | 16.14% |
| | Without PR | 1,686.38 | | 85.08% | 73.03% | 26.79% | 15.55% | 25.97% | **16.12%** |

Note: 'PR' stands for 'path relinking'.

### 5.5. Effect of the number of vehicles and percentage of willingness-to-share

In the previous section, we assume that all passengers are willing to share a vehicle with others. The percentage of willingness-to-share means the proportion of passengers who want to take ride-hailing sharing services in the total number of passengers who want to take ride-hailing services. When the value of the percentage of willingness-to-share is 50%, half of the passengers want to share the vehicles, while the other half of passengers only want to ride alone

without sharing. In this section, the length of time interval used in the simulation is 60 s. The passengers who want to share a vehicle are randomly chosen from all requests when the percentage of willingness-to-share is 50%. Moreover, when additional candidate vehicles are introduced into this experiment, the origins of those additional vehicles are determined randomly according to the strategy described in Section 5.1. As shown in Tables 6 and 7, the objective value and the matching percentage increase when the number of vehicles increases and vice versa. However, the upward trend becomes slow when the number of vehicles is larger than 10,000. When the percentage of willingness-to-share decreases, both the objective value and the matching percentage decrease. Moreover, when the number of vehicles is large, the matching percentage slightly increases as the percentage of willingness-to-share increases. Table 8 shows that the sharing percentage decreases with the decreasing percentage of willingness-to-share and the increasing number of vehicles. It demonstrates that few vehicles for ride-hailing services can promote ride-hailing sharing. Overall, the trends agree with our expectations.

**Table 6 Comparison of the objective value in terms of the number of vehicles and percentage of willingness-to-share**

| Objective value | | Number of vehicles | | | | |
|---|---|---|---|---|---|---|
| | | 1,000 | 2,400 | 6,000 | 10,000 | 15,000 |
| **The percentage of willingness-to-share** | 0 | 551.76 | 977.90 | 1,235.85 | 1,307.50 | 1,349.07 |
| | 50% | 807.74 | 1,265.21 | 1,465.05 | 1,522.57 | 1,562.37 |
| | 100% | 1,210.15 | 1,689.94 | 1,840.11 | 1,882.02 | 1,905.40 |

**Table 7 Comparison of the matching percentage in terms of the number of vehicles and percentage of willingness-to-share**

| Matching percentage | | Number of vehicles | | | | |
|---|---|---|---|---|---|---|
| | | 1,000 | 2,400 | 6,000 | 10,000 | 15,000 |
| **Percentage of willingness-to-share** | 0 | 47.75% | 77.68% | 89.54% | 92.13% | 93.50% |
| | 50% | 54.14% | 81.23% | 90.17% | 92.35% | 93.72% |
| | 100% | 61.49% | 85.15% | 91.51% | 93.01% | 93.99% |

**Table 8 Comparison of the sharing percentage in terms of the number of vehicles and percentage of willingness-to-share**

| Sharing percentage | | Number of vehicles | | | | |
|---|---|---|---|---|---|---|
| | | 1,000 | 2,400 | 6,000 | 10,000 | 15,000 |
| **Percentage of willingness-to-share** | 0 | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| | 50% | 36.98% | 32.58% | 31.35% | 30.46% | 30.25% |
| | 100% | 77.43% | 73.11% | 71.19% | 70.51% | 70.21% |

### 5.6. Effect of the delay coefficient

This section discusses the effect of the delay coefficient $co_{delay}$ on the performance measures introduced in Section 5.3. Table 9 compares the results with different lengths of time intervals using different values of $co_{delay}$. For all lengths of time intervals, a higher delay coefficient

achieves better results on the objective value and all performance measures except for $R_{time}$ and $RT_{time}$. It is reasonable because a larger coefficient means a larger tolerance of passengers to longer travel time, which allows the operators to have more feasible matches but leads to larger $R_{time}$ and $RT_{time}$.

**Table 9 Performance comparison in terms of delay coefficients**

| Time interval | Delay coefficient | Objective value | Matching | Sharing | $R_{money}$ | $R_{time}$ | $RT_{money}$ | $RT_{time}$ |
|---|---|---|---|---|---|---|---|---|
| 10 s | 1.3 | 1,677.33 | 85.24% | 72.10% | 26.32% | **15.48%** | 25.73% | **16.25%** |
| | 1.5 | **2,136.25** | **88.20%** | **80.12%** | **33.53%** | 23.37% | **32.17%** | 24.82% |
| 30 s | 1.3 | 1,682.97 | 85.08% | 73.28% | 26.70% | **15.56%** | 25.73% | **16.23%** |
| | 1.5 | **2,149.78** | **88.35%** | **80.35%** | **33.89%** | 23.39% | **32.64%** | 24.72% |
| 60 s | 1.3 | 1,689.94 | 85.15% | 73.11% | 26.87% | **15.51%** | 26.03% | **16.14%** |
| | 1.5 | **2,157.17** | **88.25%** | **80.72%** | **34.28%** | 23.48% | **32.83%** | 24.73% |

### 5.7. Analysis of the objective function

In this paper, there are three components considered in the objective function, including the number of served customers, the travel cost ratio, and the travel time ratio. Either a too small travel cost ratio or a too large travel time ratio can prevent customers from selecting ride-hailing sharing services. To illustrate the importance of considering travel time and cost ratios in the objective function, two new objective functions are introduced. The first one excludes the travel time ratio by setting $b_3$ as zero, while keeping $b_1$ and $b_2$ unchanged. The second one excludes the travel cost ratio by setting $b_2$ as zero, while keeping $b_1$ and $b_3$ unchanged.

As shown in Table 10, when the travel time ratio is not considered in the objective function, the average travel time increment percentage per passenger and the percentage of total travel time increment increase significantly. The ride-hailing matching trips allow long detours to serve customers. However, long detours lower the allowable number of additional passengers served in the later time intervals due to the fixed time window of passengers aboard, leading to the reduction in the sharing percentage and thus the reduction in $R_{money}$ and $RT_{money}$. When the travel cost ratio is not considered in the objective function, the sharing percentage, the average out-of-pocket cost saving percentage per passenger, and the percentage of total out-of-pocket cost saving decrease significantly because ride-hailing sharing requires vehicles to detour to pick up customers and increase the travel time ratio. Meanwhile, the reduction in the sharing percentage induces the decrease in $R_{time}$ and $RT_{time}$ because fewer ride-hailing sharing activities imply fewer detours experienced by the passengers. Therefore, both the travel time ratio and travel cost ratio are important components in the objective function to achieve better ride-hailing sharing services.

**Table 10 Comparison of different objective functions**

| Time interval | Objective Function | Objective value | Matching | Sharing | $R_{money}$ | $R_{time}$ | $RT_{money}$ | $RT_{time}$ |
|---|---|---|---|---|---|---|---|---|
| 10 s | Normal | 1,677.33 | 85.24% | 72.10% | 26.32% | 15.48% | 25.73% | 16.25% |
| | TIME- | 4,131.83 | 85.91% | 68.65% | 23.75% | 19.98% | 23.56% | 19.83% |
| | COST- | 3,904.86 | 86.07% | 35.86% | 9.75% | 9.89% | 7.06% | 9.06% |
| 30 s | Normal | 1,682.97 | 85.08% | 73.28% | 26.70% | 15.56% | 25.73% | 16.23% |
| | TIME- | 4,142.12 | 85.88% | 68.89% | 24.16% | 19.88% | 23.76% | 19.71% |
| | COST- | 3,908.42 | 86.10% | 35.47% | 9.51% | 9.78% | 6.74% | 9.02% |
| 60 s | Normal | 1,689.94 | 85.15% | 73.11% | 26.87% | 15.51% | 26.03% | 16.14% |
| | TIME- | 4,149.36 | 85.82% | 69.54% | 24.51% | 19.78% | 24.33% | 19.64% |
| | COST- | 3,897.30 | 85.77% | 34.04% | 8.94% | 9.61% | 6.49% | 8.87% |

Note: Normal = the objective function with the three components mentioned in Equation (1), TIME- = the objective function with the number of matched requests and the travel cost ratios only, COST- = the objective function with the number of matched requests and the travel time ratios only.

### 5.8. Comparison to GRASP with path relinking

The performance of the proposed method is compared to GRASP with path relinking proposed by Santos and Xavier (2015). There are four substantial differences between the MABC algorithm of this paper and their method (named *GRASP*). First, the adopted main algorithm is different (i.e., the ABC algorithm versus GRASP). Second, the MABC algorithm incorporates the VP tree to narrow the search range for the requests. Due to the adoption of the VP tree, the solution initialization methods between the MABC algorithm and *GRASP* are different, in which *GRASP* uses the greedy method to match vehicles with feasible requests, while the MABC algorithm uses the greedy method to match requests with feasible vehicles (described in Section 4.4). Third, this paper introduces a transfer operator in addition to the swap operator that has been adopted in *GRASP*. Fourth, *GRASP* allows new requests to be added only after the destination point of the last delivered passenger aboard if there are passengers aboard the vehicle at the current time, while the MABC algorithm has no such restriction (e.g., Figure 1). The MABC algorithm can be viewed as the solution method obtained by introducing the four major modifications to *GRASP*. To have a fair comparison of the performance between *GRASP* and the MABC algorithm, the parameter setting for *GRASP* is determined based on the strategy presented by Santos and Xavier (2015) and the dataset mentioned in Section 5.1.

To clearly illustrate the effects of introducing each modification to *GRASP* on solving the studied problem, three additional new methods, which are the variants of either the MABC algorithm or *GRASP*, are proposed. Table 11 describes these variants. Each method in Table 11 only has one difference compared with its adjacent method.

**Table 11 Comparison of different methods solving the dynamic ride-hailing sharing problem**

| Method | Main algorithm | Initialization Method | Operators | Insertion restriction |
|---|---|---|---|---|
| *GRASP* (Santos and Xavier, 2015) | GRASP | Vehicle | Swap | Yes |
| *GRASP+* | GRASP | Vehicle | Swap | No |
| *ABC--* | ABC | Vehicle | Swap | No |
| *ABC-* | ABC | Vehicle | Swap + transfer | No |
| *MABC* | ABC | Request + VP | Swap + transfer | No |

tree

Note: Vehicle = initialization method using the greedy method to match vehicles with feasible requests; Request = initialization method using the greedy method to match requests with feasible vehicles; Swap = swap operator, Transfer = transfer operator.

As shown in Table 12, the method using the MABC algorithm proposed in this paper (*MABC*) performs the best. The variants of the MABC algorithm and *GRASP*, which include new features to the ABC algorithm, yield a better solution compared with *GRASP*. The comparison between *GRASP* and *GRASP+* shows that removing the insertion restriction (the fourth aspect) is an effective way to improve solution quality. It can improve the chance for ride-hailing sharing, simultaneously complying with the time window, capacity, and travel cost constraints for each request. The comparison between *GRASP+* and *ABC--* demonstrate that, without the transfer operator, GRASP achieves a better objective value in longer time intervals, and the ABC algorithm performs better in shorter time intervals. The comparison between *ABC--* and *ABC-* showed that the transfer operator greatly improves the performance. The comparison between the MABC algorithm and *ABC-* demonstrates that the modified initialization method and the VP tree used in this paper are more effective than the initialization method used in *GRASP* because the latter is time-consuming. In summary, Table 12 shows that the proposed MABC algorithm performs well in solving the dynamic ride-hailing sharing problem and that the modifications to *GRASP* are effective.

Table 13 shows the performance of the MABC algorithm and GRASP in terms of the objective function adopted by Santos and Xavier (2015), who consider the number of matched requests and travel cost ratio. The results demonstrate that the sharing percentage increases significantly, and the customers can save more money when using the proposed MABC algorithm, which also leads to an increment in the objective value. Comparing Table 12 with Table 13, it can be observed that the customers need to waste more travel time to finish trips when ignoring the travel time ratio in the objective function.

**Table 12 Performance comparison between different variants of the MABC algorithm and GRASP**

| Time interval | Algorithm | Objective Value | Matching | Sharing | $R_{money}$ | $R_{time}$ | $RT_{money}$ | $RT_{time}$ |
|---|---|---|---|---|---|---|---|---|
| 10 s | *GRASP* | 1,211.71 | 85.22% | 17.60% | 5.93% | **12.02%** | 5.60% | **11.08%** |
| | *GRASP+* | 1,362.77 | 88.15% | 43.11% | 12.50% | 14.64% | 11.08% | 14.20% |
| | *ABC--* | 1,388.19 | **88.53%** | 44.31% | 13.25% | 14.36% | 12.22% | 13.99% |
| | *ABC-* | 1,433.01 | 88.04% | 51.10% | 15.32% | 14.62% | 14.51% | 14.36% |
| | *MABC* | **1,677.33** | 85.24% | **72.10%** | **26.32%** | 15.48% | **25.73%** | 16.25% |
| 30 s | *GRASP* | 1,274.66 | 85.66% | 22.51% | 7.71% | **10.95%** | 7.39% | **10.10%** |
| | *GRASP+* | 1,566.38 | 88.61% | 56.81% | 16.87% | 14.16% | 16.22% | 14.32% |
| | *ABC--* | 1,548.72 | **88.42%** | 57.77% | 18.68% | 14.16% | 17.39% | 14.27% |
| | *ABC-* | 1,642.22 | 88.06% | 67.93% | 22.92% | 15.11% | 22.16% | 15.50% |
| | *MABC* | **1,682.97** | 85.08% | **73.28%** | **26.70%** | 15.56% | **25.73%** | 16.23% |
| 60 s | *GRASP* | 1,292.96 | 85.30% | 23.02% | 7.89% | **10.82%** | 7.60% | **10.14%** |
| | *GRASP+* | 1,617.69 | 88.20% | 62.34% | 20.54% | 14.17% | 18.83% | 14.44% |
| | *ABC--* | 1,603.08 | **88.39%** | 62.73% | 20.58% | 14.17% | 18.86% | 14.37% |
| | *ABC-* | 1,661.39 | 87.49% | 69.25% | 24.06% | 15.26% | 23.35% | 15.86% |
| | *MABC* | **1,689.94** | 85.15% | **73.11%** | **26.87%** | 15.51% | **26.03%** | 16.14% |

**Table 13 Performance comparison between the MABC algorithm and GRASP in terms of the objective function without travel time ratio**

| Time interval | Algorithm | Objective Value | Matching | Sharing | $R_{money}$ | $R_{time}$ | $RT_{money}$ | $RT_{time}$ |
|---|---|---|---|---|---|---|---|---|
| 10 s | *GRASP* | 3,599.15 | 85.56% | 15.78% | 5.45% | 18.40% | 5.17% | 16.80% |
| | *MABC* | 4,131.83 | 85.91% | 68.65% | 23.75% | 19.98% | 23.56% | 19.83% |
| 30 s | *GRASP* | 3,688.64 | 86.38% | 21.16% | 7.38% | 18.67% | 7.54% | 17.13% |
| | *MABC* | 4,142.12 | 85.88% | 68.89% | 24.16% | 19.88% | 23.76% | 19.71% |
| 60 s | *GRASP* | 3,720.96 | 86.63% | 24.59% | 8.14% | 18.79% | 8.14% | 17.44% |
| | *MABC* | 4,149.36 | 85.82% | 69.54% | 24.51% | 19.78% | 24.33% | 19.64% |

## 6. Conclusions

In this paper, a dynamic ride-hailing sharing problem is proposed, which aims to maximize the weighted difference between the number of served customers and the sum of the travel cost ratio and travel time ratio. Meanwhile, the time window and travel cost constraints of the passengers and the capacity constraint of the vehicles are considered simultaneously. To handle the dynamic characteristics of the ride-hailing sharing problem, the problem was divided into many static subproblems with an identical time interval length. In each time interval, the request collection and matching algorithm were executed simultaneously. To solve subproblems, we propose a method based on the artificial bee colony algorithm, in which the vantage-point tree is used to narrow the search space of the algorithm and path relinking is incorporated to accelerate the solution speed to get the better solution. The method using the GRASP with path relinking proposed by Santos and Xavier (2015) was selected as the benchmark for the comparison. The results show that our proposed method outperforms the benchmark. The results also demonstrate the following. (a) With a longer time interval, the performance of the proposed method is better. However, it should be noted that a longer time interval leads to a longer time of data collection and algorithm execution, which requires the passengers to wait longer for matching results. (b) Embedding path relinking into the ABC algorithm significantly improves the performance of the resultant solution method. (c) The percentage of willingness-to-share and the number of ride-hailing vehicles can significantly influence the matching percentage and the sharing percentage of the ride-hailing sharing problem. (d) With a higher tolerance for the detouring time due to ride-hailing sharing, the proposed method can perform significantly better. (e) Considering both travel cost and travel time ratios into the design objective can achieve the best sharing percentage, and balance the increase in travel time ratio and the decrease in travel cost ratio compared with the design objectives that miss either the travel time or the travel cost ratio. (f) Ride-hailing sharing can generate benefits to the passengers as the passengers can spend less money on ride-hailing fares by spending a little bit more time due to the detours.

This study opens the following interesting future research directions. First, our solution method is simple and efficient but does not consider the lookahead policy. Introducing the lookahead policy can often improve the performance of some classical transportation systems (e.g., Mitrović-Minić et al., 2004; Spivey and Powell, 2004; Sayarshad et al., 2020; Sayarshad & Gao, 2020). Therefore, one of the future search directions is to extend our solution method to incorporate this lookahead policy. Second, in this study, we only consider the ride-hailing service offered by a private company. If the company was public operated, the taxi charge could be lower if the passengers waiting time was longer. This socially efficient price could be examined by modifying the price mechanism in the proposed formulation, which is an

interesting research direction. In the future, we can analyze socially efficient prices in a ride-hailing sharing problem similar to the studies of Figliozzi et al. (2007) and Sayarshad and Chow (2015).

## References


Agatz, N. A. H., Erera, A. L., Savelsbergh, M. W. P., & Wang, X. (2011). Dynamic ride-sharing: a simulation study in metro Atlanta. *Transportation Research Part B Methodological*, *45*(9), 1450-1464.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462-467.

Attanasio, A., Cordeau, J. F., Ghiani, G., & Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, *30*(3), 377-387.

Baugh Jr., J. W., Kakivaya, G. K. R., & Stone, J. R. (1998). Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization*, *30*(2), 91-123.

Beaudry, A., Laporte, G., Melo, T., & Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR Spectrum*, *32*(1), 77-107.

Berbeglia, G., Cordeau, J. F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*(1), 8-15.

Cordeau, J. F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, *153*(1), 29-46.

Coslovich, L., Pesenti, R., & Ukovich, W. (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, *175*(3), 1605-1615.

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*(1), 7-22.

Figliozzi, M. A., Mahmassani, H. S., Jaillet, P. (2007). Pricing in dynamic vehicle routing problems. *Transportation Science*, *41*(3), 302-318.

Fu, A. W. C., Chan, P. M. S., Cheung, Y. L., & Moon, Y. S. (2000). Dynamic VP-tree indexing for N-nearest neighbor search given pair-wise distances. *The VLDB Journal*, *9*(2), 154-173.

Geisberger, R., Sanders, P., Schultes, D., & Delling, D. (2008). Contraction hierarchies: Faster and simpler hierarchical routing in road networks. *In International Workshop on Experimental and Efficient Algorithms* (pp. 319-333). Springer, Berlin, Heidelberg.

Geisberger, R., Sanders, P., Schultes, D., & Vetter, C. (2012). Exact routing in large road networks using contraction hierarchies. *Transportation Science*, *46*(3), 388-404.

Glover, F. (1997). Tabu search and adaptive memory programming—Advances, applications and challenges. In *Interfaces in Computer Science & Operations Research* (pp. 1-75). Springer, Boston, MA.

Glover, F., Laguna, M., & Marti, R. (2000). Fundamentals of scatter search and path relinking. *Control & Cybernetics*, *29*(3), 653-684.

Ho, S. C., Szeto, W. Y., Kuo, Y. H., Leung, J. M., Petering, M., & Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, *111*, 395-241.

Horn, M. E. (2002). Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies*, *10*(1), 35-63.

Intergovernmental Panel on Climate Change (IPCC) (2015). *Climate Change 2014: Mitigation*

*of Climate Change* (Vol. 3). Cambridge University Press.
Jaw, J. J., Odoni, A. R., Psaraftis, H. N., & Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, *20*(3), 243-257.
Jensen, P. (2005). *Indicator: Occupancy Rates of Passenger Vehicles*. Technical Report, European Environmental Agency. Retrieved from https://www.eea.europa.eu/data-and-maps/indicators/occupancy-rates-of-passenger-vehicles/occupancy-rates-of-passenger-vehicles.
Jung, J., Jayakrishnan, R., & Park, J. Y. (2016). Dynamic shared-taxi dispatch algorithm with hybrid-simulated annealing. *Computer-Aided Civil and Infrastructure Engineering*, *31*(4), 275-291.
Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization* (Vol. 200). Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.
Karaboga, D., & Ozturk, C. (2009). Neural networks training by artificial bee colony algorithm on pattern classification. *Neural Network World*, *19*(3), 279-292.
Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of the Franklin Institute*, *346*(4), 328-348.
Liang, X., de Almeida Correia, G. H., An, K., & van Arem, B. (2020). Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times. *Transportation Research Part C: Emerging Technologies*, *112*, 260-281.
Long, J., Szeto, W. Y., & Huang, H. J. (2014). A bi-objective turning restriction design problem in urban road networks. *European Journal of Operational Research*, *237*(2), 426-439.
Ma, S., Zheng, Y., & Wolfson, O. (2013). T-share: A large-scale dynamic taxi ridesharing service. In *2013 IEEE 29th International Conference on Data Engineering* (pp. 410-421). IEEE Computer Society.
Ma, S., Zheng, Y., & Wolfson, O. (2015). Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge & Data Engineering*, *27*(7), 1782-1795.
Madsen, O. B. G., Ravn, H. F., & Rygaard, J. M. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*, *60*(1), 193-208.
Melachrinoudis, E., Ilhan, A. B., & Min, H. (2007). A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, *34*(3), 742-759.
Mitrović-Minić, S., Krishnamurti, R., & Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, *38*(8), 669-685.
Nielsen, F., Piro, P., & Barlaud, M. (2009). Bregman vantage point trees for efficient nearest neighbor queries. In *IEEE International Conference on Conference: Multimedia and Expo, 2009* (pp. 878-881).
Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, *14*(2), 130-154.
Resendel, M. G., & Ribeiro, C. C. (2005). GRASP with path-relinking: Recent advances and applications. In *Metaheuristics: Progress as Real Problem Solvers* (pp. 29-63). Springer, Boston, MA.
Santos, A., McGuckin, N., Nakamoto, H.Y., Gray, D., & Liss, S. (2011). *Summary of Travel Trends: 2009 National Household Travel Survey*. Technical Report, Federal Highway Administration, US Department of Transportation.
Santos, D. O., & Xavier, E. C. (2015). Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, *42*(19), 6728-6737.

Sayarshad, H. R., & Chow, J. Y. J. (2015). A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological*, *81*(2), 539-554.

Sayarshad, H. R., & Gao, H. O. (2018). A scalable non-myopic dynamic dial-a-ride and pricing problem for competitive on-demand mobility systems. *Transportation Research Part C: Emerging Technologies*, *91*, 192-208.

Sayarshad, H. R., & Gao, H. O. (2020). Optimizing dynamic switching between fixed and flexible transit services with an idle-vehicle relocation strategy and reductions in emissions. *Transportation Research Part A: Policy and Practice*, *135*, 198-214.

Sayarshad, H. R., Mahmoodian, V., & Gao, H. O. (2020). Dynamic non-myopic routing of electric taxis with battery swapping station. *Sustainable Cities and Society*, *57*, 102113.

Schilde, M., Doerner, K. F., & Hartl, R. F. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, *238*(1), 18-30.

Schrank, D., Eisele, B., Lomax, T., & Bak, J. (2015). *2015 Urban Mobility Scorecard.* Technical Report, Texas A&M Transportation Institute.

Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, *9*(2), 625-631.

Spivey, M. Z., & Powell, W. B. (2004). The dynamic assignment problem. *Transportation Science*, *38*(4), 399-419.

Szeto, W. Y., & Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, *215*(1), 126-135.

Szeto, W. Y., & Jiang, Y. (2014). Transit route and frequency design: Bi-level modeling and hybrid artificial bee colony algorithm approach. *Transportation Research Part B: Methodological*, *67*(9), 235-263.

Szeto, W. Y., & Shui, C. S. (2018). Exact loading and unloading strategies for the static multi-vehicle bike repositioning problem. *Transportation Research Part B: Methodological*, *109*, 176-211.

Wang, Y., Zheng, B., & Lim, E. P. (2018). Understanding the effects of taxi ride-sharing—A case study of Singapore. *Computers, Environment and Urban Systems*, *69*, 124-132.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 311-321). Society for Industrial and Applied Mathematics.