Two-Phase Scheduling for Efficient Vehicle Sharing

Ji Liu, Carlyna Bondiombouy, Lei Mo, Patrick Valduriez

Two-phase Scheduling for Efficient Vehicle Sharing

Ji Liu, *Member, IEEE,* Carlyna Bondiombouy, Lei Mo, *Member, IEEE,* and Patrick Valduriez

***Abstract*—Cooperative Intelligent Transport Systems (C-ITS) is a promising technology to make transportation safer and more efficient. Ridesharing for long-distance is becoming a key means of transportation in C-ITS. In this paper, we focus on private long-distance ridesharing, which reduces the total cost of vehicle utilization for long-distance journeys. In this context, we investigate journey scheduling problem with shared vehicles to reduce the total cost of vehicle utilization. Most of the existing works directly schedule journeys to vehicles with long scheduling time and only consider the cost of driving travellers instead of the total cost. In contrast, to reduce the total cost and scheduling time, we propose a comprehensive cost model and a two-phase journey scheduling approach, which includes path generation and path scheduling. On this basis, we propose two path generation methods: a simple near optimal method and a reset near optimal method as well as a greedy based path scheduling method. Finally, we present an experimental evaluation with different path generation and path scheduling methods with synthetic data generated based on real-world data. The results reveal that the proposed scheduling approach significantly outperforms baseline methods in terms of total cost (up to 69.8%) and scheduling time (up to 84.0%) and the scheduling time is reasonable (up to 0.16s). The results also show that our approach has higher efficiency (up to 141.7%) than baseline methods.**

***Index Terms*—Vehicle sharing, Path planning problem, Scheduling, Optimization**

I. Introduction

Cooperative Intelligent Transport Systems (C-ITS) [1] promise to make transportation safer and more efficient. By integrating different data sources, C-ITS can deeply change people's driving experience by reducing traffic congestion, providing intelligent transportation algorithms, reducing significantly the number of traffic accidents, and finally realizing unmanned vehicles [2]. For instance, by integrating the location of different vehicles, a centralized system could generate a smart scheduling plan for all the related vehicles. However, vehicle locations in different agencies are not frequently updated or the data are not fully exploited. When there is an order, an agency's employees typically assign the order to an available vehicle in the agency without paying attention to the overall cost. In addition, there is no cooperation among different vehicle agencies and thus, no data integration. By integrating the data about vehicle locations in different agencies in a single system, we could achieve smart scheduling so that the total cost of vehicle utilization is reduced.

J. Liu (e-mail: liuji04@baidu.com) is with the Big Data Laboratory, Baidu Research, 100085 Beijing, China.

C. Bondiombouy (e-mail: carlynablessing@gmail.com) is with African Center of Excellence in Mathematical Sciences, IT and Applications, 01 BP 613 Porto-Novo, Benin.

L. Mo (e-mail: lmo@seu.edu.cn) is with the School of Automation, Southeast University, 210096 Nanjing, China.

P. Valduriez (e-mail: patrick.valduriez@inria.fr) is with Inria, University of Montpellier, CNRS, LIRMM, 34095 Montpellier, France.

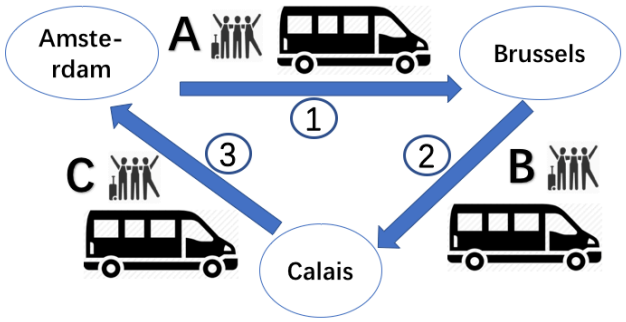


Fig. 1: An example for long-distance ridesharing. A, B, C represent three groups of travellers distributed in Amsterdam, Calais and Brussels. 1, 2, 3 respectively represent Group A travels from Amsterdam to Brussels, Group B from Brussels to Calais and Group C from Brussels to Amsterdam.

Ridesharing is a promising approach for reducing energy consumption and traffic congestion while satisfying people's commuting needs [3]. Ridesharing based on private cars, often known as carpooling or recurring ridesharing, has been studied for years to deal with people's routine commutes. Furthermore, ridesharing based on buses or other vehicles for long-distance travelling is becoming a promising transport manner in C-ITS [4]. According to the analyses of the French National Transport Survey [5] and the Europe Environment Agency (EEA) Report [6], the long-distance is defined as the distance longer than 80 km across different cities or countries. In this paper, we focus on private long-distance ridesharing with vehicles, where several vehicles can realize a set of journeys for different groups of travellers while a vehicle can only realize a journey for a single group at a time in order to ensure the privacy and efficiency of each journey. Private long-distance ridesharing can be widely exploited to reuse buses from different bus agencies for several traveller groups in order to reduce the total cost of vehicle utilization.

As shown in the example presented in Figure 1, three groups of travellers (A, B and C) are respectively distributed in three cities, e.g., Amsterdam, Brussels and Calais. Group A needs to go to Brussels (1), Group B needs to go to Calais (2) and Group C needs to go to Amsterdam (3). Assume timing is perfect, i.e., just after Group A arrives in Brussels (1), Group B can start travelling and just after Group B arrives in Calais (2), Group C can begin to go to Amsterdam (3). In this case, without private long-distance ridesharing, each group needs to rent a vehicle to go from one city to another with high cost, which includes the cost to return back the vehicle from their destination city to the original city. With private long-distance ridesharing, only one vehicle needs to be exploited, which will go from Amsterdam to Brussels, Brussels to Calais, and finally back to Amsterdam. This simple example shows a significant cost reduction for three groups

with one vehicle. Furthermore, significant cost can be reduced by exploiting private long-distance ridesharing for multiple groups of travellers and vehicles.

In order to provide private long-distance ridesharing, the scheduling of journeys to each vehicle is critical to satisfy travellers' requirements in order to reduce total cost. The journey scheduling process is to map each journey to a vehicle and decide if the vehicle should be returned back to its original location after realizing the journey. The result of the scheduling process is a scheduling plan. A journey can be realized by any available vehicle with different cost. We assume that the vehicle is available for driving the travellers during day time and that there is always an available driver to drive the vehicle according to the scheduling plan[1]. A vehicle has its original location and should be returned back to its original location after realizing its scheduled journeys. As the travellers are sensitive to the cost of using vehicles, the objective of journey scheduling is to reduce the total cost of realizing all the journeys in the orders. The total cost is the cost of using the vehicle, including transporting travellers, driving the vehicle to the start location of each journey and returning back to the original location. Thus, the problem we address in this paper is how to efficiently generate a scheduling plan that reduces the total cost of all the orders.

There are already some scheduling methods [7], [8], [9], [10] for ridesharing. However, they do not consider the cost to drive the vehicle to the start location of the scheduled journey or the cost to return the vehicle back to its original location. In addition, existing scheduling methods directly schedule journeys to available vehicles, which corresponds to long scheduling time. In this paper, we decompose the problem into two sub-problems and propose a two-phase journey scheduling approach, which generates paths based on the journeys and then schedules each path to vehicles in order to minimize the total cost. A path [11] is a combination of successive journeys to be scheduled as a unit to a vehicle. A vehicle is not returned back to its original location until all the journeys in the path are realized. We propose a cost model to calculate the total cost to schedule journeys to different vehicles. We propose two path generation methods, i.e., Simple Near Optimal (SNO) and Reset Near Optimal (RNO), which yield low cost. In addition, we propose a greedy path scheduling method in order to schedule each path to a vehicle in order to reduce the total cost. The two-phase approach, SNO and the greedy path scheduling method have already been implemented in the scheduling system of a startup company, i.e., TUDING[2]. The main contributions of this paper are:

1) A problem formulation of journey scheduling problems. We decompose the journey scheduling problem into two sub-problems, i.e., path generation problem and path scheduling problem.
2) A cost model for journey scheduling. The cost model calculates the total cost to realize all the journeys corresponding to a scheduling plan with the consideration of the cost to drive the vehicles to the start location of each journey and the cost to return vehicles.
3) A two-phase journey scheduling approach with two path generation algorithms and a greedy path scheduling algorithm. The two-phase journey scheduling approach first generates paths based on journeys and then schedules each journey to vehicles in order to reduce the total cost based on the cost model.
4) An extensive experimental evaluation on synthetic data based on real-world data, that shows the advantages of our approach, compared with baseline algorithms.

This paper is organized as follows. Section II discusses related work. Section III defines some basic concepts, proposes a cost model and formally defines the scheduling problem we address in this paper. Section IV describes our proposed two-phase journey scheduling approach including path generation methods and path scheduling methods. Section V presents an experimental evaluation based on synthetic data generated from real-world data, which shows the advantages of our proposed methods. Finally, section VI concludes.

II. Related Work

Many existing journey scheduling approaches directly schedule each journey to each vehicle using heuristics in order to minimize a cost function [7], [8]. These approaches do not combine the journeys into paths and the corresponding search space is big (see details in Section III). Lam *et al.* [7] propose a distributed genetic algorithm for scheduling. The principle of a genetic algorithm is to encode possible scheduling plans into a population of chromosomes, and subsequently to transform the population using standard operations of selection, crossover and mutation, producing successive generations, until the convergence condition is met. It is hard to configure the initial chromosomes and convergence condition when there is no additional information about the scheduling. In addition, strict convergence condition can incur long execution time as many iterations are needed [12]. Greedy scheduling algorithms are proposed to maximize the profits of taxi drivers [9], while the total cost of the journeys is not considered. The journeys can be adjusted in order to reduce the number of vehicles to use for the journeys [8], [10]. However, these methods are not used for reducing the total cost to realize the journeys. The total cost includes the cost to drive the vehicle to the start location of each journey and the cost to return the vehicle to its original location. A heuristic solution is proposed to schedule journeys to public vehicles within a short distance, e.g., a city, in order to have big sharing rate within short scheduling time [11]. However, this solution does not address the long-distance vehicle sharing and does not consider the cost to return the vehicles to the original location. It also directly schedules journeys to vehicles without generating paths. In addition, some approaches have other objectives than the total cost, e.g., reducing the number of buses [13], maximizing the number of bus riders [14], improving the utilization efficiency of buses

[1]We assume that a driver is always available for a vehicle. Under the given scheduling plan, we assign a driver to a vehicle. If the driver is not available for the whole path, more drivers can be assigned to the vehicle to ensure that this vehicle can be driven according to the scheduling plan. The cost of sending a driver to vehicle is out of the scope of the paper as this depends on the preference of the drivers.

[2]https://www.tudingbus.com/

[15], or optimizing the traffic based on behavior prediction [16], [17].

In the case of long-distance journeys, the cost of vehicles from different agencies or countries is different. Thus, the cost of different scheduling plans is different. In addition, the cost to return the vehicle may be significant. As a result, it is critical to share a rented vehicle for a long time in order to reduce the cost using a scheduling algorithm. In the case of short distance journeys, the vehicles are generally within the same city. Thus, the cost of different vehicles is generally similar. In addition, the cost to return the vehicle may be negligible. In this case, there is no need to rent a vehicle for a long time. Thus, the problem of long-distance vehicle sharing is different from that of short-distance.

Scheduling vehicles to journeys is similar to the scheduling of parallel tasks in a distributed system, which is an NP-hard problem [18]. The vehicles resemble the distributed servers while each journey corresponds to a task to be scheduled. There are many algorithms to schedule tasks in distributed systems, e.g., Opportunistic Load Balancing (OLB) [19], Minimum Completion Time (MCT) [19], Data-Intensive Multi-site task scheduling (DIM) [15] and Heterogeneous Earliest Finish Time (HEFT) [20]. After adapting them to the scheduling process of vehicles, they can be described as follows.

OLB randomly selects an available or an earliest available vehicle for a journey while MCT schedules a journey to the vehicle that can finish its previous journey first with the consideration of the time to move from its last place to the start point of the journey. DIM first schedules the journeys to the vehicle that has the same place as the start point of the journey. Then, it balances the workload in each vehicle in order to achieve load balancing and reduce the overall execution cost. HEFT gives the priority to each journey according to the dependencies between journeys and the distance of the journey, i.e., the distance between the start location and the end location of the journey. Then, it schedules the journeys with the highest priority to the vehicle that can finish the journey first. However, the aforementioned scheduling algorithms do not take the cost of moving the vehicles from one place to the start location of a journey or the cost of returning the vehicle to its original location into consideration, which is critical to vehicle sharing. The ActGreedy scheduling algorithm [12] can be used to efficiently schedule the journeys to vehicles for reducing the cost based on a cost model. However, it does not consider grouping different journeys into paths in order to reduce the total cost to realize all the journeys.

The problem we address can be formulated as the Mixed-Integer Linear Program (MILP) problem as explained in III-E. Although the B&B method [21] can provide an optimal solution for the MILP problem, the complexity of B&B can be exponential [22], which is unacceptable. In our work, our proposed two-phase approach first generates paths and then schedules paths to different vehicles. A basic approach to generating paths is to encapsulate each journey as a path. However, this approach cannot optimize the combination of different journeys to reduce cost. A-star is an efficient algorithm to generate a shortest path between two points [23]. However, it lacks the sensibility of the total cost of the generated path. We propose a greedy approach to generate a path, which corresponds to the minimal total cost based on a cost model. Inspired by [8], we propose the adjustment of journeys while meeting the time requirements of the journeys in order to reduce the total cost. In addition, there are different scheduling algorithms to match each path to different vehicles. A basic approach for path scheduling is to randomly select a vehicle for each path. Based on the location of the vehicle, a second approach can be to schedule the path to the bus, which has the nearest location to the start place of the path. This approach can reduce the cost to drive the vehicle from its original location to the start location of the path. In addition, a path can also be scheduled to the available vehicle that has the lowest renting cost. However, this approach does not consider the total cost to realize the whole paths. In this paper, we propose a greedy approach, which selects the vehicle that corresponds to the minimum cost to realize the generated paths.

When vehicle utilization is high, the corresponding scheduling plan typically leads to journey delay problems. A delay on one journey can cascade through the schedule, delaying all subsequent journeys on that path, and if all vehicles are in use there are no spares in case of breakdowns, crashes, or severe delays. A common strategy to deal with it is to have some buffer time between journeys and paths [24] while the buffer may cause more cost [25]. In our cost model, we can set the rest time (see details in Formula 10) longer to address the journey delay. As the execution time of our algorithm (RNO or SNO) is very short, we can re-execute the scheduling algorithm when there is an important delay. And then, we dynamically change the scheduling plan based on the new results of re-execution. In addition, we can assign a new vehicle to the impacted path without changing other scheduling plans of other paths [26]. Note that the last two strategies can be also used in a railway system: a train that is late has impact on other trains (because of connections) and the schedule needs to be adjusted.

III. Problem Definition

In this section, we define some basic concepts, present the time constraints and give a motivating example. Then, we propose a cost model to estimate the cost corresponding to a scheduling plan and finally formulate the problem we address in this paper.

A. Notations and Definitions

We first introduce some important terms, e.g., order, journey and journey scheduling.

Let us assume that there are multiple requirements from travellers, denoted by orders, and diverse vehicles to be used to achieve these orders. An **order** represents that a group of travellers wants to travel to different locations. We assume that the number of travellers in each group is smaller than a limit, e.g., 20 travellers, so that one vehicle is able to pick up all the travellers. If an order has more travellers than the limit, the order can be split into multiple orders and the number of travellers in each split order is no more than the limit. An

order contains multiple journeys. A **journey** represents the smallest unit of travel, which should be realized by a vehicle to transport a group of travellers from a start location to an end location with a time duration defined by the start time and end time of the journey. The time period of the start time and end time should be long enough for a vehicle to travel from the start location to the end location. The start time of a journey can be postponed or the end time can be brought forward, i.e., the group of travellers arrive at the end location before the original end time. A vehicle has an original location, which means that the vehicle is retrieved from that location and should be finally returned back to it. The cost of using a vehicle includes renting, driving without passengers and driving with passengers. The cost of renting the vehicle is per day. The cost to drive without passengers is the cost to drive the vehicle to the start location of each journey or the cost to return the vehicle to its original location. The cost to drive with passengers is the cost to transport the travellers from the start location to the end location defined in the journey. When the vehicle is at its original location, the cost to drive without passengers is null. The vehicles belong to different types, and different types correspond to different costs.

A vehicle can realize several journeys from different orders without being returned, and thus the vehicle can realize several paths in order to realize the related journeys. As defined in Section I, a **path** [11] is a combination of successive journeys to be scheduled as a unit to a vehicle. All the journeys of a path are realized by a vehicle in a sequential manner without returning the vehicle back to its original location. At the end of each path, the vehicle should be returned back to its original location. In addition, a vehicle can realize multiple paths sequentially. For instance, after it is returned after realizing all the journeys in a path, a vehicle can start realizing the journeys of another path.

Journey scheduling is the process that maps each journey to a vehicle, which generates a scheduling plan. The time to generate a scheduling plan is called scheduling time. A scheduling plan defines which vehicle will realize which journey and whether the vehicle is returned back to its original location after realizing the journey.

We summarize the symbols used in this paper in Table I.

TABLE I: Symbols used in our cost model.

Symbol	Meaning
J	Set of journeys (indexed by i or k)
V	Set of vehicles (indexed by l)
P	Set of paths (indexed by r or u)
SP	A scheduling plan defined by a $n * m$ matrix
$sp_{i,l}$	Scheduling decision for journey j_i, vehicle v_l
n	Number of journeys in J
w	Number of vehicles in V
m	Number of paths in P
stj_i	Start time of journey j_i
etj_i	End time of journey j_i
stp_r	Start time of path p_r
etp_r	End time of path p_r
J_l	Set of journeys scheduled to vehicle v_l
P_l	Set of paths scheduled to vehicle v_l
slj_i	Start location of journey j_i
elj_i	End location of journey j_i
ol_l	Original location of vehicle v_l
slp_r	Start location of path p_r
elp_r	End location of path p_r
$t_2 - t_1$	The amount of time between t_1 and t_2
loc_i	A location with the index i

B. Time Constraints

A scheduling plan should meet three time constraints.

1) The first one (Formula 1) is that different scheduled journeys should not have overlapping time periods if they are assigned to the same vehicle. For instance, journey j_i is scheduled to vehicle v_l. Then, the start time stj_k and end time etj_k of another journey j_k that is scheduled to vehicle v_l should meet the following constraint: stj_i is later than etj_k or etj_i is earlier than stj_k, which is defined in Formula 1 with ">" representing later than and "<" representing earlier than.

$$\begin{aligned} &\forall\ two\ journeys\ j_i\ \in\ J_l,\ j_k\ \in\ J_l, \\ &\qquad stj_i\ >\ etj_k\ or\ etj_i\ <\ stj_k \end{aligned} \tag{1}$$

2) The second time constraint (Formula 2) is that the intervals of any two successive journeys in the same path should be long enough to move the vehicle from the end location of the former journey (j_i) to the start location of the later journey (j_k). Formula 2 ensures that there is enough time between two journeys, e.g., journeys j_i and j_k, in the same path to drive the vehicle from the end location and the end time of journey j_i to the start location of journey j_k before the start time of journey j_k. In Formulas 2 and 3, function $Time$ represents the amount of time that it takes to drive the vehicle from one location to another.

$$\begin{aligned} &\forall\ two\ successive\ journeys\ j_i,\ j_k\ \in\ path, \\ &\qquad stj_k - etj_i\ >\ Time(el_i, sl_k) \end{aligned} \tag{2}$$

3) The third time constraint (Formula 3) is that the intervals of any two successive paths scheduled to the same vehicle should be long enough to return the vehicle from the end location of the former path to its original location and to drive the vehicle from its original location to the start location of the following path.

$$\begin{aligned} &\forall\ two\ successive\ paths\ p_r,\ p_u\ \in\ P_l, \\ &\qquad stp_u - etp_r\ >\ Time(elp_r, ol_l) + Time(ol_l, slp_u) \end{aligned} \tag{3}$$

C. A Motivating Example

Let us assume that there are m vehicles distributed at different locations, n journeys ordered by travellers, and a scheduling plan. When a set of journeys is scheduled to a vehicle, the scheduled journeys and corresponding paths should meet the time constraints given in Section III-B. The vehicle should drive the travellers from the start location to the end location of each journey. We assume that a vehicle is capable of transporting the travellers from the start location to the end location within the required time of each journey. A path starts from an earliest journey that does not need to return the vehicle and ends at a journey that needs to return the vehicle back to its original location. The vehicle is driven back to its original location after realizing the last journey

in a path. The last journey is the journey that has the latest end time in all journeys of the path. Afterwards, the vehicle continues realizing the following paths until all its scheduled paths are realized.

The total cost of a vehicle is the cost to realize all the paths. The cost to realize a path is the cost of each journey and that of returning the vehicle to its original location (when the vehicle is rented for a short time, e.g., one day, there is some extra cost incurred for returning the vehicle at a different location than the original location [27], [28]). The cost corresponding to a journey includes the cost to drive the vehicle to the start location of the journey and the cost to transport the travellers from the start location to the end location of the journey. The time to rent a vehicle is counted from the time when the vehicle starts being driven to the start location of the first journey in the path to the time when the vehicle is returned back to its original location. The first journey is the journey that has the earliest start time in all journeys of the path. The cost to return the vehicle to its original location can be ignored when the renting time is longer than a limit, e.g., 3 months (vehicle agencies can offer a discount on the price of returning vehicles when the renting time is longer than a limit, because they earn enough money). The vehicles of different types have different original locations and may have different costs to move from one location to another.

Different scheduling plans can incur different costs. In this paper, we address the problem of how to efficiently generate a scheduling plan in order to reduce the total cost. Let us assume that there are three vehicles distributed at Amsterdam (vehicle A), Brussels (vehicle B) and Calais (vehicle C) as shown in Figure 1. If we do not consider the Cost to Return vehicles to their Original Location (CROL), then we could schedule journey 1 to vehicle A, journey 2 to vehicle B and journey 3 to vehicle C (scheduling plan sp_1). In this case, there will be the cost to return vehicle A back to Amsterdam, vehicle B back to Brussels and Vehicle C back to Calais, in reality. If we schedule the three journeys with time constraints and CROL, e.g., journeys 1, 2, 3, to the vehicle at Amsterdam (vehicle A) (scheduling plan sp_2), there will not be the cost to return the vehicle back to Amsterdam. It is obvious that sp_2 is better than sp_1 as sp_2 reduces the cost to return vehicles in reality. In fact, reusing vehicle A for journeys 2 and 3 is transforming the cost to return vehicle A after journey 1 to the cost to realize journeys 2 and 3. In this paper, we propose a two-phase journey scheduling approach to generate a scheduling plan similar to sp_2 in order to reduce the total cost.

D. Cost Model

The cost of all journeys can be calculated using Formula 4, where the total cost of all orders is composed of the cost of each vehicle with the scheduled journeys.

$$TotalCost(SP) = \sum_{v_l \in V} Cost_l(SP) \tag{4}$$

In Formula 4, SP represents a scheduling plan, which is defined by a matrix of scheduling decision variables, i.e., a matrix of $sp_{i,l}$. $TotalCost(SP)$ represents the cost of the scheduling plan SP. $Cost_l(SP)$ represents the cost of vehicle v_l according to the scheduling plan SP. $j \rightarrow v$ represents that vehicle v is assigned to journey j. The decision variable $sp_{i,l}$ is defined as

$$sp_{i,l} = \begin{cases} 1, \ if \ j_i \ \rightarrow \ v_l \ with \ returning \ v_l \\ -1, \ if \ j_i \ \rightarrow \ v_l \ without \ returning \ v_l \\ 0, \ if \ j_i \ is \ not \ scheduled \ to \ v_l \end{cases} \tag{5}$$

The total cost (Formula 6) of a vehicle is composed of the total cost of all the paths of the vehicle. In Formula 6, $PathCost_{r,l}(SP)$ represents the total cost to realize Path p_r using vehicle v_l according to scheduling plan SP.

$$Cost_l(SP) = \sum_{p_r \in P_l} PathCost_{r,l}(SP) \tag{6}$$

1) Cost of A Path: The total cost of a path (Formula 7) is composed of the cost to realize all the scheduled journeys, the time cost to use the vehicle for each path and the cost to return the vehicle from the end location of the last journey to its original location. In Formula 7, $\sum_{j_i \in p_r} \sum_{v_i \in V} journeyCost_{i,l} * |sp_{i,l}|$ represents the cost to realize all the journeys in p_r according to SP. $journeyCost_{i,l}$ represents the cost to realize j_i using v_l. If j_i is scheduled to v_l, $|sp_{i,l}|$ is 1. $TimeCost_{r,l}(SP)$ represents the renting cost to use the vehicle for the path. The renting time starts from the time to drive the vehicle to the start location of the first journey of the path and ends at the time when the vehicle is returned to its original location. $TimeCost_{r,l}(SP)$ can be calculated based on Formula 11. $ReturnCost_{r,l}(SP)$ represents the cost to return the vehicle from the end location of its last scheduled journey in p_r to the original location of v_l. When the renting time of a vehicle for the path is longer than a limit, $ReturnCost_{r,l}(SP)$ can be ignored and the total cost of a path can be calculated based on Formula 8.

$$\begin{aligned} PathCost_{r,l}(SP) = & \sum_{j_i \in p_r} \sum_{v_l \in V} journeyCost_{i,l} * |sp_{i,l}| \\ & + TimeCost_{r,l}(SP) \\ & + ReturnCost_{r,l}(SP) \end{aligned} \tag{7}$$

$$\begin{aligned} PathCost_{r,l}(SP) = & \sum_{j_i \in p_r} \sum_{v_i \in V} journeyCost_{i,l} * |sp_{i,l}| \\ & + TimeCost_{r,l}(SP) \end{aligned} \tag{8}$$

2) Cost of A Journey: The cost to realize a journey (Formula 9) is composed of the cost to drive the vehicle to the start location of the journey and the cost to drive the travellers from the start location to the end location of the journey. In Formula 9, $journeyCost_{i,l}$ represents the cost to realize the journey j_i using vehicle v_l. $idleDrivingCost_l$ represents the cost to drive the vehicle without passengers per time period, e.g., hour. $Time(Loc(v_l), slj_i)$ represents the time to drive the vehicle to the start location of a journey. $Loc(v_l)$ represents the location of vehicle v_l. The location of the vehicle is determined by whether j_i is the first journey in a path. If it is the first journey, the location of vehicle v_l is the original location of v_l. Otherwise, the location is the end location of the previous journey that is scheduled to v_l. This can be calculated based on SP. $drivingCost_l$ represents the cost to drive the vehicle with a group of passengers per time period, i.e., hour. $Time(slj_i, elj_i)$ represents the time to drive the vehicle with the group of passengers from the start location of j_i to the end location of j_i, which can be calculated based on Formula 10.

$$journeyCost_{i,l} = idleDrivingCost_l * Time(Loc(v_l), slj_i) + drivingCost_l * Time(slj_i, elj_i) \quad (9)$$

In Formula 10, $Distance(loc_i, loc_j)$ represents the distance to drive vehicle v from one location to another. $averageSpeed$ is configured by the user of the scheduling algorithm. The calculation of the time considers that the vehicle can be driven in the daily time and that the driver should have a rest every three hours. This is applied in the $AdaptTime$ function of Formula 10. In addition, we assume that there is always an available driver to drive the vehicle according to a scheduling plan. The distance between two locations can be calculated based on Baidu maps [29], Google maps [30] or formulas of orthodromic distance [31], which is outside of the scope of this paper.

$$Time(loc_i, loc_j) = AdaptTime(\frac{Distance(loc_i, loc_j)}{averageSpeed}) \quad (10)$$

3) Renting Cost of A Path: The renting cost of a path is calculated based on Formula 11. $rentingCost_l$ represents the cost to rent v_l per day. $PathTime(p_r)$ represents the time of path p_r, which is calculated based on Formula 12 or 13.

$$TimeCost_{r,l}(SP) = rentingCost_l * PathTime(p_r) \quad (11)$$

Formula 12 represents the time of a path, which is composed of the time for the journeys in the path, the time to drive the vehicle to the start location of the first journey and the time to return the vehicle. The time for the journeys in the paths starts from the start time of the first journey, e.g., j_i, to the end time of the last journey, e.g., j_k. The time to drive the vehicle to the start location of the first journey and the time to return the vehicle can be calculated based on Formula 10. When the renting time for the journeys ($etj_k - stj_i + Time(ol_l, slj_i)$) is longer than a time limit, e.g., 3 months, the time to return the vehicle can be ignored and the time of a path can be calculated based on Formula 13.

$$PathTime(p_r) = etj_k - stj_i + Time(ol_l, slj_i) + Time(elj_k, ol_l) \quad (12)$$

$$PathTime(p_r) = etj_k - stj_i + Time(ol_l, slj_i) \quad (13)$$

4) Cost to Return: The cost to drive the vehicle from the end location of the last journey of a path to its original location is calculated based on Formula 14. $ReturnCost_{r,l}(SP)$ represents the cost to drive the vehicle from its end location of the last journey of path p_r to its original location. In Formulas 14 and 9, $idleDrivingCost_l$ represents the cost to drive Vehicle v_l without passengers per time period, e.g., hour. $Time(elp_r, ol_l)$ represents the time to drive v_l from the end location of p_r to its original location. The end location of p_r is the end location of the last journey in p_r, which can be calculated based on SP. $Time(elp_r, ol_l)$ can be calculated based on Formula 10.

$$ReturnCost_{r,l}(SP) = IdleDrivingCost_l * Time(elp_r, ol_l) \quad (14)$$

E. Problem Formulation

The scheduling problem we address in this paper is how to schedule each journey to an available vehicle in order to minimize the total cost for all journeys. The problem is formulated as: how to calculate a decision variable matrix SP in order to

$$\min_{S.T.} TotalCost(SP) \quad s.t. \begin{cases} (1), (2), (3) \\ sp_{i,l} \in \{1, -1, 0\}, \forall sp_{i,l} \in SP \end{cases} \quad (15)$$

We call this problem the original scheduling problem. Each decision variable contains three possible choices. The size of the search space of this problem is very large. We can reduce this search space by decomposing the problem into two sub-problems, i.e., the decomposed problem. The first sub-problem, i.e., path generation problem, is how to group the journeys in a small number of paths in order to minimize the cost of all journeys. The second problem, i.e., scheduling problem, is how to schedule the paths to vehicles in order to minimize the total cost.

1) Path Generation Problem: The first sub-problem is how to calculate a path generation plan in order to achieve the objective defined in Formula 16. $TotalCost(PGP)$ represents the total cost to realize the journeys in a set of paths, which does not contain the cost to return back the vehicles to the original locations. PGP represents a path generation plan. The time cost to use the vehicle from the start time of a path to the end time of a path is calculated based on Formula 18.

$$\min_{S.T.} TotalCost(PGP) \quad s.t. \begin{cases} (1), (2) \\ pgp_{i,r} \in \{0, 1\}, \forall\ pgp_{i,r} \in PGP \end{cases} \quad (16)$$

where PGP is defined by a matrix of binary decision variables $pgp_{i,r}$, and $pgp_{i,r}$ is defined in Formula 17.

$$pgp_{i,r} = \begin{cases} 1\ if\ j_i \in p_r \\ 0\ otherwise \end{cases} \quad (17)$$

PGP can be represented by a set of paths P. The total cost of P is calculated based on Formula 18. $journeyCost_{i,l}$ represents the cost to realize a journey. We assume there is a vehicle v_l, whose $idleDrivingCost_l$, $drivingCost_l$ and $rentingCost_l$ is the average cost of all the available vehicles. Although v_l is a vehicle with fixed parameters, we keep index l in order to have consistency with Formula 9. $PathTime(p_r)$ represents the time of renting the vehicle, which is calculated based on Formula 13.

$$TotalCost(PGP) = \sum_{p_r \in P} (\sum_{j_i \in p_r} journeyCost_{i,l} * pgp_{i,r} + rentingCost_l * PathTime(p_r)) \quad (18)$$

2) Path Scheduling Problem: The second sub-problem is how to schedule each path to a vehicle in order to minimize the total cost, which is given in Formula 19. PSP represents the scheduling plan of paths, which is defined by a matrix of binary scheduling decision variables $psp_{r,l}$ defined in Formula 20. $TotalCost(PSP, PGP)$ represents the total cost of all paths, which is calculated based on Formula 21.

$$\min_{S.T.} TotalCost(PSP, PGP) \quad s.t. \begin{cases} (3) \\ psp_{r,l} \in \{0, 1\}, \forall\ pspr, l \in PSP \end{cases} \quad (19)$$

The binary scheduling decision variables $psp_{r,l}$ are defined as follows.

$$psp_{r,l} = \begin{cases} 1\ if\ p_r\ is\ scheduled\ to\ v_l \\ 0\ otherwise \end{cases} \quad (20)$$

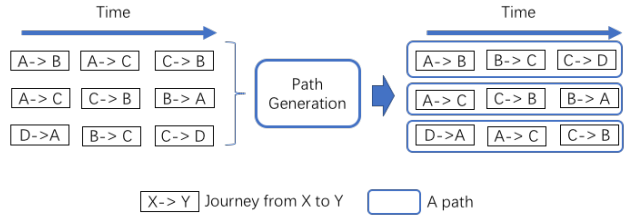


Fig. 2: Path generation. Journeys are combined to generate paths. Each journey belongs to only one path. The journeys in the same path can only be sequentially realized.

As defined in Formula 21, $TotalCost(PSP, PGP)$ is the sum of the costs to realize each path.

$$TotalCost(PSP, PGP) = \sum_{p_r \in P} \sum_{v_l \in V} PathCost_{r,l}(getSP(PSP, PGP)) * psp_{r,l} \tag{21}$$

where $PathCost_{r,l}(getSP(PSP, PGP))$ represents the cost to realize p_r using v_l. The function $getSP(PSP, PGP)$ to infer the scheduling plan SP used in Formula 7 or 8 is based on Formula 22. Then, the cost of each path can be calculated based on Formula 7 or 8.

$$sp_{i,l} = \begin{cases} 1, \ if \ pgp_{i,r} = 1 \ and \ psp_{r,l} = 1 \ and \ j_i \ is \ not \\ \ the \ last \ journey \ in \ p_r \\ -1, \ if \ pgp_{i,r} = 1 \ and \ psp_{r,l} = 1 \ and \ j_i \ is \ the \\ \ last \ journey \ in \ p_r \\ 0, \ if \ pgp_{i,r} = 0 \ or \ psp_{r,l} = 0 \end{cases} \tag{22}$$

These two sub-problems have a linear objective function and inequality constraints. Some of the variables are binary, e.g., path generation plan and path scheduling plan, while the rest are real. Thus, the two sub-problems are Mixed-Integer Linear Program (MILP) problems [32]. These problems are not easy to solve since the search space is complex and this space increases exponentially with the number of journeys, the number of vehicles and the number of paths.

3) Search Space Analysis: Let us assume that there are n journeys, w vehicles and m paths. The size of the search space of the original scheduling problem is 3^{w*n}. The size of the search space of the decomposed problem is $2^{w*m} * m^n$, where 2^{w*m} is the size of the search space to schedule paths to vehicles and m^n is the size of the search space to generate paths. In most cases, e.g., the cases except Baseline presented in Section V, the size of the search space of the original scheduling problem is bigger than that of the decomposed problem according to Lemma III.1.

Lemma III.1. *The search space of the decomposed problem is smaller than that of the original scheduling problem when* $m <= n$ *and* $m < (3/2)^w$.

Proof: When $m < (3/2)^w$, *we have* $2^w * m < 3^w$. *Then,* $(2^w * m)^n < (3^w)^n$, *which equals to* $2^{w*n} * m^n < 3^{w*n}$. *When* $m <= n$, *we have* $2^{w*m} <= 2^{w*n}$. *Thus,* $2^{w*m} * m^n < 3^{w*n}$. ■

IV. Two-Phase Journey Scheduling

We propose two-phase journey scheduling to address the journey scheduling problem defined in Formula 15. This approach first groups the journeys to generate paths as shown in Figure 2 and then schedules the generated paths to vehicles as shown in Figure 3. Grouping the journeys to paths addresses the first sub-problem defined in Formula 16 and scheduling the generated paths to vehicles addresses the second sub-problem defined in Formula 19. Algorithm 1 describes the process of two-phase journey scheduling. In this algorithm, Line 1 first generates paths based on the journeys while achieving the minimum total cost. When grouping the journeys, we use an average cost of all the buses to calculate the cost of each path. Line 2 schedules different paths to different vehicles in order to achieve a small total cost.

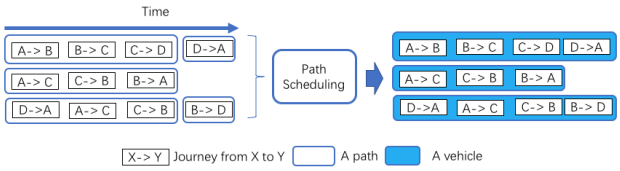


Fig. 3: Path scheduling. Each path is scheduled to a vehicle. The scheduled paths of the same vehicle should meet the time constraints presented in Section III-B.

Algorithm 1 Journey scheduling

Input: $journeys$: a set of journeys; V: a set of vehicles
Output: SP: a scheduling plan that assigns each journey to a vehicle
$paths \leftarrow GeneratePaths(journeys)$
$SP \leftarrow SchedulePathsToVehicles(paths, V)$

We can use different methods to generate paths and schedule paths to vehicles, which we present in this section. The calculation of cost is based on Formula 21 and the cost model presented in Section III-D.

A. Path Generation

In this section, we adapt A-Star [33] for path generation and propose two methods to generate paths, i.e., Simple Near Optimal (SNO) and Reset Near Optimal (RNO). In addition, we take a baseline method to compare our proposed approaches. The baseline path generation algorithm simply encapsulates each journey to a path.

1) Adapted A-Star: Inspired by [33], the A-star algorithm can be used to search for the shortest path from a start point to an end point. We can adapt the A-star algorithm to generate paths based on a set of journeys, as shown in Algorithm 2.

The goal of our Adapted A-Star is to generate paths in order to reduce the total cost. Algorithm 2 has two loops. Lines 2-16 generate all the possible paths based on the journeys. Line 4 gets the earliest start time in the journeys that are not added to generated paths, i.e., the earliest start time in all the journeys. Line 5 gets the start location corresponding to the earliest start time in Line 4. Lines 6-14 add all the journeys to a path. Line 7 selects the suitable journey that requires the smallest additional cost to be added to the path according to Formula 18. In addition, the selected journey should meet the second time constraint defined in Formula 2 (see details in Section III-A). The start time of the selected journey should be late enough so that there is enough time to drive the vehicle from the end location of the last journey in the path to the

start location of the selected journey before the start time of the selected journey. The time constraint is verified in the $getSmallestCost$ function. If the interval between the end time of the path and the start time of the selected journey is longer than three days, we stop adding new journeys (Lines 8-10). Line 11 adds the journey to the path. Lines 12-13 update the end time and the start location of the path.

Algorithm 2 Adapted A-Star

Input: J: a set of journeys
Output: P: a set of paths
1: $P \leftarrow \emptyset$
2: **while** not all journeys in J are added in a path **do**
3: $path \leftarrow \emptyset$
4: $endTime = earliestStartTime(J)$
5: $startLocation = firstStartLocation(J)$
6: **while** at least a journey in J can be added to $path$ **do**
7: $journeyToAdd \leftarrow getSmallestCost(endTime, startLocation, J)$
8: **if** $Interval(journeyToAdd, path) > limit$ **then**
9: break
10: **end if**
11: $path \leftarrow path \cup journeyToAdd$
12: $endTime = endTime(journeyToAdd)$
13: $startLocation = endLocation(journeyToAdd)$
14: **end while**
15: $P \leftarrow P \cup path$
16: **end while**

2) Near Optimal Algorithm: Inspired by the ActGreedy scheduling algorithm [12], we propose a near optimal algorithm to generate a set of paths, which is presented in Algorithm 3. The idea of this algorithm is to reuse a vehicle when there is a journey, whose start location is in the surrounding of the destination of the previous journey in the near future, in order to transform the cost to return vehicles to the cost of realizing other journeys so that the total cost is reduced.

Algorithm 3 Near Optimal Algorithm

Input: J: a set of journeys
Output: P: a set of paths
1: $P \leftarrow \emptyset$
2: $sortAccordingToStartTime(J)$
3: **for** Each journey in J **do**
4: $path \leftarrow getPathWithSmallestCost(P, journey)$
5: $path \leftarrow path \cup journey$
6: **end for**
7: **for** Each path in P **do**
8: **for** Each journey in $path$ **do**
9: **if** The cost can be reduced by splitting the path **then**
10: $P \leftarrow merge(P, split(path, journey))$
11: **end if**
12: **end for**
13: **end for**

In Algorithm 3, we first sort the journeys according to their start time (Line 2). Then, for each journey, we assign it a path that corresponds to the smallest additional cost, i.e., the difference between the total cost after adding the journey and that before adding the journey (Lines 3-6). In Line 4, if there is no available path in P to add journey $journey$, an empty path is created and returned. In addition, the newly created path is put into P in the $getPathWithSmallestCost$ function. Then, for each path, we split it into multiple paths in order to reduce the total cost (Lines 8-12) by reducing the useless renting time. The journeys in a path are in ascending order of start time, i.e., $path$ = $\{j_1, j_2,..., j_i, j_{i+1}, ..., j_n\}$ with the start time of j_i earlier than that of j_{i+1}. When splitting a path at j_i (Function $split$), the path is split into $path_1$ and $path_2$ with $path_1$ = $\{j_1, j_2,..., j_i\}$ and $path_2$ = $\{j_{i+1}, ..., j_n\}$. $path_1$ is returned and put into P and the original path ($path$) is replaced by $path_2$ in the $merge$ function (Line 10).

While adding a journey to the path (Line 4), we have two situations. The first situation is that the start time and the end time of the journey cannot be modified. In this situation, we directly add the journey to the path. We call this method **Simple Near Optimal** (**SNO**). The second situation is that we can adjust the start time and the end time of the journey J_i in order to reduce the total cost for the following journeys. We call this adjustment method **Reset Near Optimal** (**RNO**). RNO can postpone the start time or bring the end time forward, i.e., $stj_i' >= stj_i$ or $etj_i' <= etj_i$, in order to reduce the cost according to a temporary path generation plan. Note that as the original time period includes the adjusted time period for a journey, the travellers should value the adjusted journey times as highly as the original one. After adjustment, the journey should be able to be realized within the start time and the end time from the start location to the end location as defined in Formula 23. In Formula 23, $etj_i' - stj_i'$ represents the time period of J_i, after adjustment, and $Time(slj_i, elj_i)$ represents the time to drive the travellers from the start location of the journey to the end location of the journey, which can be calculated based on Formula 10.

$$etj_i' - stj_i' \geq Time(slj_i, elj_i) \tag{23}$$

3) Path Generation Algorithm Complexity: Let n denote the number of journeys. The complexity of Baseline is $\mathcal{O}(n)$. The complexity of Adapted A-Star is $\mathcal{O}(n^2)$ (the complexity of Function $getSmallestCost$ is $\mathcal{O}(n)$ as it needs to iterate on all the remaining journeys). This complexity is much bigger than the baseline approach and corresponds to long scheduling time. The complexity of SNO and RNO depends on the sorting algorithm, which can be $\mathcal{O}(n \log n)$ using TimSort [34], [35]. This complexity is much smaller than that of Adapted A-Star and is slightly bigger than that of Baseline while it corresponds to small total cost as presented in Section V.

B. Path Scheduling

Once the paths are generated, they can be scheduled to different vehicles with small total cost. In this section, we present three basic methods, i.e., Random [36], Nearest [9] and Economic [9], and propose one greedy schedule method. When a path is scheduled to a vehicle, all the journeys in the path are scheduled to the vehicle. In addition, when a path is scheduled to a vehicle, the vehicle is able to take all the travellers in each journey of the order.

TABLE II: Parameters in average case (A). Costs are in Euro. Number represents the number of vehicles. Cost represents the cost to rent a vehicle per day. Idle cost (ICost) represents the cost to drive the vehicle without passengers per hour. Driving cost (DCost) represents the cost to drive the vehicle with a group of passengers per hour in Euro.

Type	Cost	Location	Number	ICost	DCost
1	350	Porto	2	5	10
2	370	Porto	2	5	10
3	300	Lodz	2	5	10
4	330	Warsaw	2	5	10
5	290	Athens	1	5	10
6	500	Paris	100	5	10

The Random method schedules each path to a random available vehicle after sorting the paths according to their total cost based on an average cost of all the vehicles, which is similar to OLB. The Nearest method schedules each path to a vehicle that has the nearest location to its start location after sorting. This method tries to reduce the cost to drive the scheduled vehicle to its start location. The Economic method schedules the path to the vehicle that has the smallest renting cost. This method intends to reduce the cost for long paths. However, it does not consider the overall total cost.

We propose a greedy path scheduling (Greedy) algorithm to schedule each path to a vehicle in order to reduce the total cost, which is described in Algorithm 4. Line 2 sorts the paths according to their total cost based on an average cost of all the vehicles. Then, for each path, Line 4 chooses the vehicle that takes the smallest total cost to realize the path according to Formula 21.

Algorithm 4 Greedy Path Scheduling

Input: P: a set of paths; V: a set of vehicles
Output: PSP: a path scheduling plan
1: $psp_{r,l} = 0 \ for \ \forall \ psp_{r,l} \in PSP$
2: $sortAccordingToTotalCost(P)$
3: **for** Each p_r in P **do**
4: $\quad v_l \leftarrow getVehicleWithSmallestCost(V, p_r)$
5: $\quad psp_{r,l} = 1$
6: **end for**

The complexity of Random is $\mathcal{O}(m)$, with m representing the number of paths. The complexity of Nearest and Economic is the same, i.e., $\mathcal{O}(m * w)$ with w representing the number of types of vehicles. Although the complexity of Nearest and Economic is bigger than Random, they can generate better scheduling plans as presented in Section V. The complexity of Greedy is $\mathcal{O}(m \log m)$ (when $\log m$ is bigger than w) or $\mathcal{O}(m * w)$ (when $\log m$ is smaller than w), which is the same (for $\mathcal{O}(m * w)$) or slightly bigger (for $\mathcal{O}(m \log m)$) than the other scheduling methods while generating a better scheduling plan with small total cost (see Section V for details).

V. Experimental Evaluation

In this section, we present an experimental evaluation of our approach. We first present the experimental setup. Then, we show the experiments. Finally, we summarize the main experimental results.

A. Experimental Setup

The vehicles and the journey information is synthetically generated based on real-world data from a bus sharing startup, i.e., TUDING. We consider a dataset with 649 journeys and the vehicles originally distributed at five locations, i.e., Porto, Lodz, Warsaw, Athens and Paris. There are at most 46 journeys that cover the same time period. This means that at least 46 vehicles can realize all the 649 journeys.

For setting the parameters of the vehicles, we consider three cases, i.e., Average (A), Limited (L) and General (G). In case A, vehicles of different types have different renting costs, original locations and numbers of vehicles while other parameters, i.e., idle cost and driving cost, are the same. In case L, vehicles of different types have different renting costs and different original locations while other parameters, i.e., number of vehicles, idle cost and driving cost, are the same. In case G, vehicles differ in all the parameters, i.e., number of vehicles, location, idle cost and driving cost.

We implemented the algorithms in Java, i.e., Baseline, Adapted A-Star (AA-Star for short), SNO and RNO as path generation method; Random, Nearest, Economic and Greedy as path scheduling method. In addition, we implemented three one-phase algorithms, i.e., ActGreedy (1-Greedy) [12], Genetic [7] and Brute force. 1-Greedy and Genetic are designed without the consideration of the cost to return vehicles while we implement our cost model in these two algorithms.

The experiments are carried out in a workstation with Intel quad-core i7-3610QM CPU and 32GB RAM. We use the cost model presented in Section III-D to compute the total cost.

B. Experimental Results

In this section, we first present the experiments to show that our proposed approaches are near optimal. Then, we give the detailed results of executing the program with the 649 journeys in case A. Afterwards, we give the total cost, scheduling time and efficiency of the program execution in cases L and G.

1) Near Optimal: In order to show that the scheduling plans generated by our proposed methods are near optimal, we compare the total cost of SNO/Greedy and RNO/Greedy with the optimal scheduling plans generated by a brute force method without modifying start time and end time. As the execution time of the brute force method is very long, we only compare the results with up to 14 journeys and 1 vehicle of Types 1-5 and 2 vehicles of Type 6 as presented in Table II. As shown in Figure 4, RNO/Greedy can outperform the optimal scheduling plans without adjusting start time or end time. SNO/Greedy can generate the scheduling plans that perform as well as that of the optimal one, in many cases (42.9%). In all the cases, the difference of the total cost between SNO/Greedy and the optimal scheduling plan is less than 1.6%.

2) Average Case: In this section, we present the experiment results based on the parameters in Table II. We show the total cost, scheduling time and average efficiency of different scheduled plans generated by different methods. When using RNO, 431 journeys have been forced to modify their start or end time with 339 modifications in start time and 431 modifications in end time.

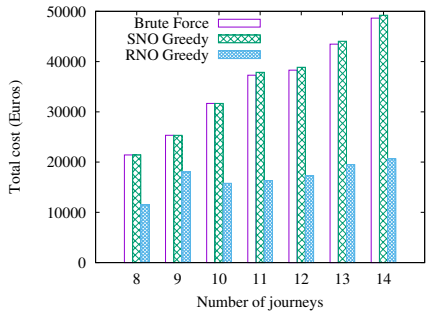


Fig. 4: Comparison with optimal scheduling plans.

The total cost calculated based on Formulas 21 is shown in Figure 5a. Random has the highest total cost compared with the other three scheduling methods. The combination of our adapted and proposed methods, i.e., AA-Star and Greedy, SNO and Greedy and RNO and Greedy, significantly outperforms (by up to 24%, 26% and 62%, respectively) the combination of Baseline and Random. Our proposed scheduling method, i.e., Greedy, performs up to 19.0%, 19.4% and 5.4% better compared with Random, Nearest and Economic, respectively. This is expected as our scheduling methods consider the total cost to realize the journeys. AA-Star outperforms Baseline up to 17.6%. SNO and RNO perform better than AA-Star and much better than Baseline. While generating paths, AA-Star, SNO and RNO reduce the total cost. In addition, RNO can significantly reduce total cost compared with SNO since it can further adjust the start time and the end time of the journey in order to reduce total cost. As RNO and SNO optimize the generated paths based on total cost, which is different from the time interval between the scheduled journeys in AA-Star, RNO and SNO have smaller total cost compared with AA-Star. When the system is not able to modify the start or end time of journeys, SNO can be used, performing up to 20.7% and 3.8% better compared with Baseline and AA-Star in terms of total cost of the generated scheduling plan. When the system is able to modify the start and end time of journeys, RNO can be used, performing up to 58.8% and 50.4% better in terms of total cost of the generated scheduling plan compared with Baseline and AA-Star. The total cost of Baseline and Greedy is already smaller than that of 1-Greedy and Genetic. In addition, our proposed approaches, i.e., RNO/Greedy and SNO/Greedy, significantly outperform 1-Greedy (18.3% and 58.4%) and Genetic (20.9% and 59.7%), in terms of total cost.

Figure 5b shows the number of generated paths. Since Baseline encapsulates each journey into a path, it yields the highest number of paths. By adapting the journeys according to the ongoing scheduling plan, RNO generates the smallest number of paths. AA-Star, SNO and RNO yield up to 60%, 71% and 84% fewer paths than Baseline.

Figure 5c shows the total scheduling time, including the time for path generation and path scheduling. The scheduling time of 1-Greedy is acceptable (0.62 seconds) while the scheduling time of genetic is very long (more than 47 seconds). Figure 5c shows that Baseline is much longer (up to 1.85, 13.25 and 16.1 times) than AA-Star, SNO and RNO, respectively. This is reasonable since Baseline generates more paths compared with the other methods. Although RNO takes time to reset the start or end time of journeys, it has shorter total scheduling time (up to 20%) since it generates fewer paths than SNO. The total scheduling time of Nearest and Economic is longer (up to 112% and 98%) than that of Random as they have slightly higher complexity. The total scheduling time of Greedy is longer than that of Random (up to 136%), Nearest (up to 60%) and Economic (45%) while its scheduling plan incurs the smallest total cost. Compared with the combination of Baseline/Random, the combination of SNO/Greedy and RNO/Greedy is respectively up to 77.3% and 78.7% better in terms of total scheduling time. The total scheduling time of Greedy with Baseline, SNO (up to 0.17s) and RNO (up to 0.16s) is less than 0.2 seconds, which is quite acceptable.

Figure 5d shows the time to generate paths, i.e., path generation time, according to different approaches. AA-Star takes the longest time to generate paths as its complexity is the highest (see Section IV-A1). The path generation time of AA-Star is up to 5.3 and 5.1 times longer than SNO and RNO, respectively. Baseline has the shortest time to generate paths as it has the smallest complexity. Although the path generation time of AA-Star, SNO and RNO is much longer than Baseline, it is much less than 1s, which is quite acceptable.

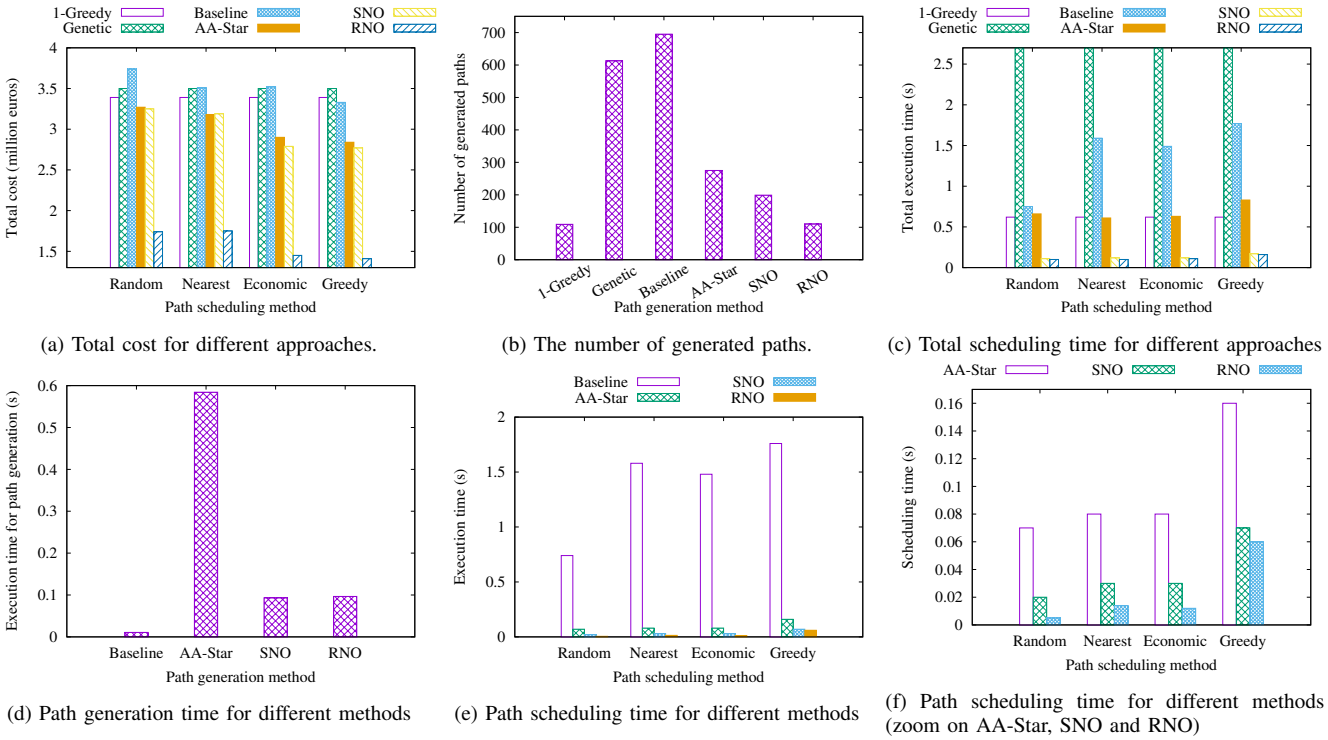
Figure 5e shows that the path scheduling time of Baseline is much longer than AA-Star (up to 19 times), SNO (up to 52 times) and RNO (up to 147 times) as Baseline schedules large number of paths. Figure 5f zooms on the path scheduling time of AA-Star, SNO and RNO. As AA-Star generates more paths than SNO and RNO, it takes more time to schedule paths. RNO yields the smallest number of paths, and its scheduling time is the smallest among all path generation methods. As the complexity of Random is the smallest, it yields the smallest path scheduling time among all path scheduling methods. The path scheduling times of Nearest and Economic are similar. The scheduling time of Greedy is longer than Random (up to 11 times), Nearest (up to 3 times) and Economic (up to 4 times) as it has higher complexity.

Furthermore, we analyse the average efficiency of different approaches. The efficiency of a scheduling plan is the average efficiency of all the vehicles, defined in Formula 24. The efficiency of a vehicle is the ratio between the driving time ($drivingTime(v_l, SP)$) and the total time of using the vehicle ($TotalTime(v_l, SP)$), as defined in Formula 25. In Formulas 25 24, the scheduling plan SP contains the path generation plan and the path scheduling plan. As defined in Formula 26, the driving time of a vehicle is the sum of the driving time of each scheduled journey. In Formula 26, $Time(slj_i, elj_i)$ represents the time to drive the passengers for a journey, which can be calculated based on Formula 10. The total time of using a vehicle is the sum of the times of all the paths scheduled to it, which can be calculated based on Formula 27. In Formula 27, $PathTime(p_r)$ can be calculated using Formula 12.

$$efficiency(SP) = \frac{\sum_{v_l \in V} efficiency(v_l, SP)}{getNumberOfVehicles(SP)} \tag{24}$$

$$efficiency(v_l, SP) = \frac{drivingTime(v_l, SP)}{TotalTime(v_l, SP)} \tag{25}$$

$$drivingTime(v_l, SP) = \sum_{p_r \in P_l} \sum_{j_i \in p_r} Time(slj_i, elj_i) \tag{26}$$



(a) Total cost for different approaches.

(b) The number of generated paths.

(c) Total scheduling time for different approaches

(d) Path generation time for different methods

(e) Path scheduling time for different methods

(f) Path scheduling time for different methods (zoom on AA-Star, SNO and RNO)

Fig. 5: Scheduling time for different approaches

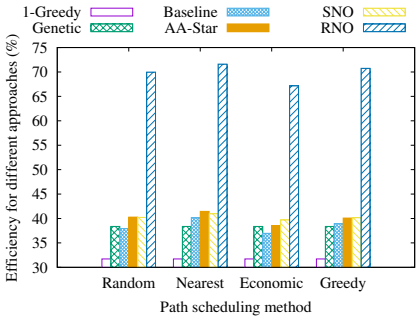


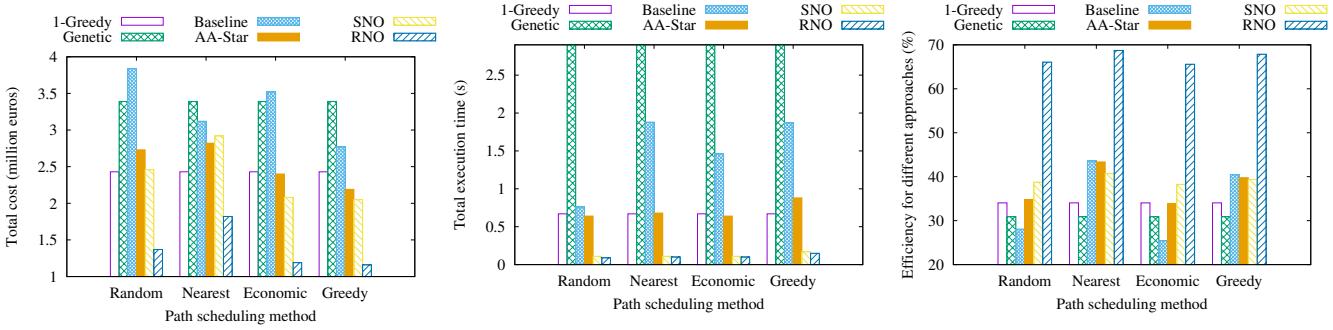
Fig. 6: Average efficiency of different approaches

$$TotalTime(v_l, SP) = \sum_{p_r \in P_l} PathTime(p_r) \tag{27}$$

Figure 6 shows the efficiency of different scheduling approaches. The efficiency of 1-Greedy is much lower than other methods (less than 32%). The combination of Baseline and Greedy outperforms both 1-Greedy and Genetic. The efficiency of different path scheduling methods is almost similar (the difference is less than 8%) while it differs much for different path generation methods. Random corresponds to the least efficiency, as it does not have optimization in terms of total cost. Our proposed path generation methods, i.e., AA-Star, SNO and RNO, have higher efficiency (up to 6.2%, 7.4% and 79.5%) compared with Baseline. Note that RNO corresponds to the highest efficiency, as it optimizes the journeys according to the scheduling plan. Compared with the combination of Baseline/Random, the combination of SNO/Greedy and RNO/Greedy is up to 3.1% and 79.3% better, respectively.

3) Limited and General Cases: In this section, we present the results in cases L and G. In case L, the vehicles of different types only differ in the renting cost and location, as shown in Table II. The parameters of the vehicles (e.g., the number of vehicles, the idle and the driving cost of different vehicles) are the same among different types of vehicles. They are set to 20, 5 and 10, respectively. Figure 7a shows the combination of RNO/Greedy and SNO/Greedy outperforms the combination of Baseline/Random up to 46.6% and 69.8%, 1-Greedy (14.3% and 51.8%) and Genetic (37.5% and 64.9%) in terms of total cost. In addition, as shown in Figure 7b, the scheduling time of RNO/Greedy and SNO/Greedy is up to 77.6% and 80.3%, respectively, smaller than Baseline/Random. The efficiency of RNO/Greedy and SNO/Greedy is respectively up to 40.2% and 141.7% higher than the combination of Baseline/Random, as shown in Figure 7c. When using RNO, 431 journeys have been forced to modify their start or end time with 336 modifications in start time and 431 modifications in end time.

In case G, the vehicles of different types differ in number, idle cost and driving cost. The numbers of Types 1-6 are respectively 5, 10, 15, 20, 25, 30; the idle costs of Types 1-6 are respectively 3, 4, 5, 6, 7, 8; the driving costs of Types 1-6 are respectively 8, 9, 10, 11, 12, 13. Figure 8a shows the combination of RNO/Greedy and SNO/Greedy outperforms the combination of Baseline/Random up to 43.9% and 68.4%, 1-Greedy (15.6% and 52.3%) and Genetic (39.5% and 65.8%) in terms of total cost. As shown in Figure 8b, the scheduling time of RNO/Greedy and SNO/Greedy is up to 81.5% and 84.0%, respectively, smaller than Baseline/Random. The efficiency of RNO/Greedy and SNO/Greedy is up to 23.3% and 123.4%, respectively, higher than Baseline/Random, as shown in Figure 8c. When using RNO, 431 journeys have been forced to modify their start or end time, where 339 modifications in start time and 431 modifications in end time. As we use average cost of all the available vehicles for the path generation, which is similar for cases A, L, and G, the numbers of journeys modified in RNO are similar.

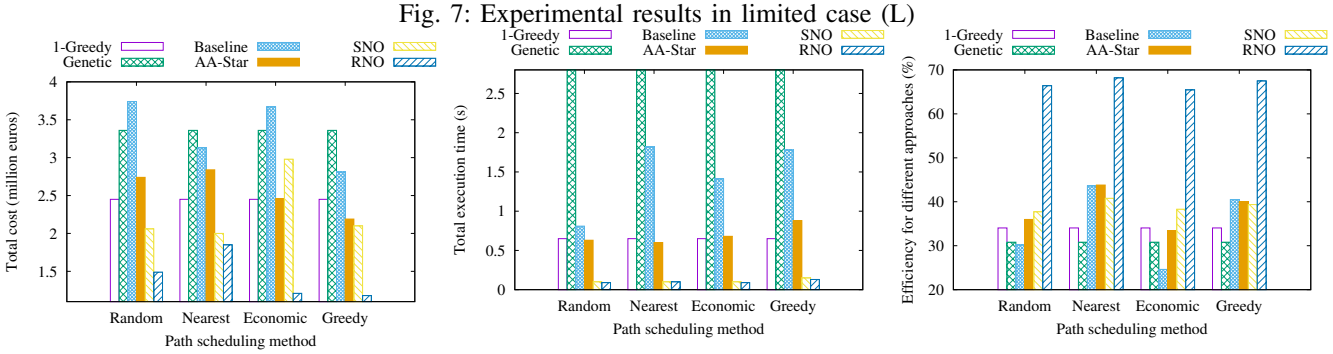


(a) Total cost for different approaches in case L

(b) Scheduling time for different approaches in case L

(c) Average efficiency of different approaches in case L

Fig. 7: Experimental results in limited case (L)



(a) Total cost for different approaches in case G

(b) Scheduling time for different approaches in case G

(c) Average efficiency of different approaches in case G

Fig. 8: Experimental results in case G

C. Concluding Remarks

While generating paths, we calculate the total cost based on the average cost of all the vehicles, which is different from the cost of the scheduled vehicle of each journey. Thus, the final result may not be an optimal solution. However, the combination of SNO or RNO and Greedy can generate a near optimal scheduling plan (the difference is less than 1.6% compared with the brute force method), which significantly outperforms the baseline method and state of the art methods, i.e., 1-Greedy and Genetic, in terms of total cost, scheduling time and efficiency. Our proposed approach, i.e., the combination of SNO/Greedy and RNO/Greedy outperforms the combination of Baseline/Random in terms of total cost (up to 46.6% and 69.8% respectively), scheduling time (up to 81.5% and 84.0% respectively) and efficiency (up to 40.2% and 141.7% respectively).

VI. Conclusion

Efficient scheduling approaches are promising to improve people's driving experience by reducing significantly the cost of using vehicles to realize travellers' orders. In this paper, we proposed a two-phase journey scheduling approach to generate scheduling plans while reducing total cost. This approach includes a cost model to calculate the total cost based on a scheduling plan, two path generation methods, i.e., SNO and RNO, and a greedy scheduling method. We evaluated our approach by comparing the proposed path generation methods to a baseline method and the scheduling method to three other methods, i.e., Random, Nearest and Economic. Our evaluation shows that our proposed path generation methods, i.e., SNO and RNO, outperform (up to 20.7% and 58.8%) Baseline in terms of total cost. In addition, Greedy performs (up to 19.0%) better than Random. The scheduling time of our approach is smaller than 2.5 seconds, which is acceptable. Our proposed path generation algorithms, i.e., SNO and RNO, correspond to higher efficiency (up to 7.4% and 79.5%) compared with Baseline. As a result, the combination of SNO/Greedy and RNO/Greedy significantly outperforms the combination of Baseline/Random in terms of total cost (up to 46.6% and 69.8% respectively), scheduling time (up to 81.5% and 84.0% respectively) and efficiency (up to 40.2%/141.7% respectively).

VII. Acknowledgement

We thank Bo Hu from TUDING for his support and help with the experimental evaluation, and Han Qiu from LTCI, Telecom Paris for fruitful discussions and useful comments. Some of the experiments in this paper were carried out on Baidu Data Federation Platform (Baidu FedCube). For platform usage, please contact via {fedcube, shubang}@baidu.com. This work was supported in part by the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China under Grant ICT20058, and in part by the Fundamental Research Funds for the Central Universities, China under Grant 2242020R10059.

References

[1] S. An, B. Lee, and D. Shin, "A survey of intelligent transportation systems," in *Int. Conf. on Computational Intelligence, Communication Systems and Networks (CICSYN)*, 2011, pp. 332–337.

[2] C. Bettisworth, M. Burt, A. Chachich, R. Harrington, J. Hassol, A. Kim, K. Lamoureux, D. LaFrance-Linden, C. Maloney, D. Perlman *et al.*, "Status of the dedicated short-range communications technology and applications: Report to congress," vol. Tech. Report, no. FHWA-JPO-15-218. U.S. Department of Transportation, Washington, DC, USA, 2015.

[3] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295 – 303, 2012.

[4] K. Augustin, R. Gerike, M. J. M. Sanchez, and C. Ayala, "Analysis of intercity bus markets on long distances in an established and a young market: The example of the u.s. and germany," *Research in Transportation Economics*, vol. 48, pp. 245 – 254, 2014.

[5] R. Gerike and A. Schulz, "Workshop synthesis: Surveys on long-distance travel and other rare events," *Transportation Research Procedia*, vol. 32, pp. 535 – 541, 2018.

[6] E. E. Agency, "Focusing on environmental pressures from long-distance transport," vol. Europe Environment Agency (EEA) Report, no. 7/2014, pp. 1–108, 2014.

[7] A. Y. S. Lam, Y. Leung, and X. Chu, "Autonomous-vehicle public transportation system: Scheduling and admission control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1210–1226, 2016.

[8] C. Tang, A. Ceder, S. Zhao, and Y. Ge, "Vehicle scheduling of single-line bus service using operational strategies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1149–1159, 2019.

[9] Y. Jia, W. Xu, and X. Liu, "An optimization framework for online ride-sharing markets," in *IEEE Int. Conf. on Distributed Computing Systems*, 2017, pp. 826–835.

[10] X. Wang, "Optimizing ride matches for dynamic ride-sharing systems," *PhD dissertation, Georgia Institute of Technology*, 2013.

[11] M. Zhu, X. Liu, and X. Wang, "An online ride-sharing path-planning strategy for public vehicle systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 616–627, 2019.

[12] J. Liu, E. Pacitti, P. Valduriez, D. de Oliveira, and M. Mattoso, "Multi-objective scheduling of scientific workflows in multisite clouds," *Future Generation Computer Systems*, vol. 63, pp. 76–95, 2016.

[13] G. Cui, J. Luo, and X. Wang, "Personalized travel route recommendation using collaborative filtering based on gps trajectories," *University Maryland, College Park*, vol. 11, pp. 284–307, 2017.

[14] Z. Wang, A. Shafahi, and A. Haghani, "Scda: School compatibility decomposition algorithm for solving the multi-school bus routing and scheduling problem," *Int. Journal of Digital Earth*, vol. 11, no. 3, pp. 284–307, 2017.

[15] Y. Liu, C. Liu, N. J. Yuan, L. Duan, Y. Fu, H. Xiong, S. Xu, and J. Wu, "Intelligent bus routing with heterogeneous human mobility patterns," *Knowledge and Information Systems*, vol. 50, no. 2, pp. 383–415, 2017.

[16] H. Slavin, Q. Yang, D. Morgan, A. Rabinowicz, J. Brandon, and R. Balakrishna, "Lane-level vehicle navigation for vehicle routing and traffic management," *U.S. Patent 9 964 414 B2*, 2018.

[17] N. O. Alsrehin, A. F. Klaib, and A. Magableh, "Intelligent transportation and control systems using data mining and machine learning techniques: A comprehensive study," *IEEE Access*, vol. 7, pp. 49 830–49 857, 2019.

[18] V. Jalaparti, P. Bodík, I. Menache, S. Rao, K. Makarychev, and M. Caesar, "Network-aware scheduling for data-parallel jobs: Plan when you can," in *ACM Conf. on Special Interest Group on Data Communication (SIGCOMM)*, 2015, pp. 407–420.

[19] M. Maheswaran, S. Ali, H. J. Siegel, D. A. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Heterogeneous Computing Workshop*, 1999, pp. 30–44.

[20] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment," *SIGMOD Record*, vol. 34, no. 3, pp. 56–62, 2005.

[21] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, no. 4, pp. 699–719, 1966.

[22] W. Zhang, "brand and bound search algorithm and their complexity," *USC/Information Sciences Institute*, vol. Tech. Report, no. ADA314598, pp. 1 – 31, 1996.

[23] F. D. n, A. Babinec, M. Kajan, P. B. no, M. Florek, T. Fico, and L. J. sica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59 – 69, 2014.

[24] M. Friedrich, M. Müller-Hannemann, R. Rückert, A. Schiewe, and A. Schöbel, "Robustness as a third dimension for evaluating public transport plans," in *Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*, 2018, pp. 4:1–4:17.

[25] M. Naumann, L. Suhl, and S. Kramkowski, "A stochastic programming approach for robust vehicle scheduling in public bus transport," *Procedia - Social and Behavioral Sciences*, vol. 20, pp. 826 – 835, 2011.

[26] O. Cats and E. Jenelius, "Planning for the unexpected: The value of reserve capacity for public transport network robustness," *Transportation Research Part A: Policy and Practice*, vol. 81, pp. 47 – 61, 2015.

[27] "Charter Bus Pricing," https://www.metropolitanshuttle.com/pricing/, 2019, [Online; accessed 27-November-2019].

[28] D. Henion, "Charter Bus Prices: How to Calculate Your Rental Costs," https://gogocharters.com/blog/charter-bus-prices/, 2019, [Online; accessed 27-November-2019].

[29] "Baidu Maps," https://map.baidu.com/, [Online; accessed 28-April-2020].

[30] "Google Maps," https://www.google.com/maps/, [Online; accessed 27-November-2019].

[31] G. Britain, "Position and direction on the earth's surface," *Admiralty Manual of Navigation*, vol. 1, pp. 1–23, 1964.

[32] C. S. Ma and R. H. Miller, "Milp optimal path planning for real-time applications," in *American Control Conference*, 2006, pp. 1–6.

[33] M. Lv, T. Gao, and N. Zhang, "Research of agv scheduling and path planning of automatic transport system," *Int. Journal of Control and Automation*, vol. 9, no. 4, pp. 1 – 10, 2016.

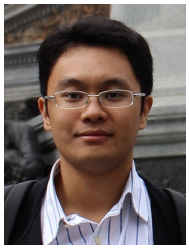
[34] P. McIlroy, "Optimistic sorting and information theoretic complexity," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993, pp. 467–474.

[35] "Sort algorithm in Java," https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort(byte[]), [Online; accessed 27-November-2019].

[36] D. Gavalas, C. Konstantopoulos, and G. Pantziou, "Design and management of vehicle-sharing systems: a survey of algorithmic approaches," in *Smart Cities and Homes*. Morgan Kaufmann, 2016, pp. 261 – 289.

Ji Liu (Member, IEEE) received the B.Sc. degree from Xidian University in 2011, the master's degree from Telecom SudParis in 2013, the Ph.D. degree from the University of Montpellier and the Microsoft Research Inria Joint Centre, Inria Zenith Team in 2016. He was a Post-Doctoral Researcher with Inria and LIRMM, University of Montpellier, France. He is currently a Researcher with the Big Data Laboratory, Baidu Research, Beijing, China. He has published several articles in international journals and conferences on scientific workflows, big data, and cloud computing. His research interests include big data, applied machine learning and distributed, and parallel data management.

Carlyna Bondiombouy received the master's degree from University Cheikh Anta Diop, Dakar, in 2013, and the Ph.D. degree in computer science from the University of Montpellier in 2017. She is currently a Researcher with the African Center of Excellence in Mathematical Sciences, IT, and Applications. Previously, she was a Big Data Consultant with Ysance, Paris. She has published several articles in international journals and conferences on big data, databases, and cloud computing. She received the best Ph.D. thesis Award from the Congolese Government in 2017. Her research interests include data science, big data, and polystores.



Lei Mo (Member, IEEE) received the B.S. degree from the Lanzhou University of Technology, China, in 2007, and the Ph.D. degree from the South China University of Technology, China, in 2013. He is currently an Associate Professor with the School of Automation, Southeast University, Nanjing, China. Previously, he was a Researcher with Inria Rennes, 1186 France, from 2017 to 2019, Inria Nancy, France, from 2015 to 2017, and Zhejiang University, China, from 2013 to 2015. His current research interests include networked control systems, cyber-physical systems, and embedded systems.

Patrick Valduriez is currently a Senior Researcher with Inria, working on distributed data management. He has authored over 300 technical articles and several textbooks, among which Principles of Distributed Database Systems. He is an ACM fellow. He was the recipient of the 1993 IBM Scientific Prize in Computer Science in France, the 2014 Innovation Award from Inria–French Academy of Science, and the Best Paper Award at VLDB 2000. He has been an Associate Editor of major journals such as VLDBJ and DAPD. He has served as a Pc Chair or a General Chair of major conferences, such as EDBT, SIGMOD, and VLDB.