Investigating Optimal Carpool Scheme by a Semi-Centralized Ride-Matching Approach

Wang Peng and Lili Du, *Member, IEEE*

***Abstract*— Carpooling service obtains a significant interest in both industry and academic fields for its potential to improve mobility and save travel costs. However, the existing approaches for exploring online carpooling routes for riders struggle with the trade-off between system optimality and computation efficiency. This study is motivated to develop a semi-centralized ride-matching strategy (SCM) to address this difficulty. Specifically, we model the carpooling service problem (CSP) by a mixed-integer programming (CSP-MIP), which explores an optimal carpooling solution for a large-scale of riders to minimize system service time by using a small CAV fleet to serve all the riders. To address the computation difficulty, we first analyze and quantify the carpooling chances (C2) among riders according to their trip features. According to the C2, we decompose a large-scale CSP-MIP involving all riders in a network into small sub-MIPs, each including a small number of riders in a carpooling community, without over sacrificing the system optimality. For each sub-MIP, we develop a network flow algorithm combined with a greedy ride-matching model to explore a local optimal solution. The experiments built upon the Hardee network demonstrated that the SCM could solve the CSP efficiently. It significantly improves the computation efficiency while maintaining the system performance as it's compared to the existing approaches.**

***Index Terms*— Carpool, community detection, mixed integer programming.**

I. Introduction

IN RECENT years, ridesharing services provided by Transportation Network Companies (TNCs), such as Uber, Lyft, and DiDi, have widely spread among populations in urban areas. They attracted significant riders thanks for providing more convenient, timely, and affordable response-on-demand services. However, some research showed that most current ridesharing services function as convenient mobile taxies with low carpool rates; ridesharing is taking away transit demand and becoming a major contributor to snarled traffic congestion and carbon emissions [26], [27]. To address this dilemma, it is critical to promote carpooling services in the ridesharing system. Moreover, it has been recognized that promoting carpooling services will benefit all aspects of stakeholders, including TNCs, riders, and society. Specifically, a high carpool rate implies that each driver will serve more riders during business hours. It will potentially reduce the number of vehicles needed for operating the system, thus reducing the operation cost. From the rider side, well-implemented carpooling services will reduce riders' travel costs while sustaining a satisfactory travel time experience. Furthermore, promoting carpooling services will also help reduce total vehicle mile traveled (VMT), fuel consumption, and emission [20], [21]. Therefore, promoting carpooling services is a promising solution for maintaining the service level (convenience) without jeopardizing traffic efficiency and sustainability.

Meanwhile, connected autonomous vehicle (CAV) technology has gained the significant interest of TNCs by its great merits, such as parking-free, long service time, low operation cost, flexible routes, and high safety resulting from removing drivers from the business [10]. Major TNCs are now investing big bills into using CAV for ridesharing services. Motivated by the above views, this study seeks to develop efficient approaches for TNCs so that they can promote their carpooling services in a CAV ridesharing system.

Current practices show that even though TNCs have provided the carpool option in their businesses, the usage is relatively low due to poor experiences. For example, Uberpool service only accounts for about 20% of all Uber trips in cities where it is available, even though pricing schemes have been implemented to encourage the users. One of the main complaints is about the travel time delay caused by the detours. These unsuccessful practices indicate that a well-accepted carpooling solution should benefit both the supply and demand sides. More exactly, to improve users' acceptance of the service, the suggested carpooling routes must not violate riders' travel time requirements. Along with these thoughts, this study seeks to help TNCs offer acceptant carpooling routes for users while minimizing the total service time of ridesharing vehicles (i.e., system service time) and the traffic overhead introduced by ridesharing vehicles.

From the methodology perspective, existing literature shows that the carpooling service problems (CSP) are often considered as the ride-matching problems involving multiple riders and multiple drivers. The corresponding mathematical models are often large-scale combinatorial problems. The high computation complexity of such problems plays a critical obstacle for their practical applications. For example, the study of [19] showed that using an exhaustive algorithm to solve a carpooling service involving 100 riders took around five and a half hours, which is too time-consuming for practical implementations.

Manuscript received 25 May 2021; revised 30 September 2021 and 20 November 2021; accepted 7 December 2021. Date of publication 11 January 2022; date of current version 12 September 2022. This work was supported in part by the National Science Foundation under Award CMMI 1818526 and Award CMMI 1817346. The Associate Editor for this article was G. Guo. *(Corresponding author: Lili Du.)*

The authors are with the Department of Civil and Coastal Engineering, University of Florida (UF), Gainesville, FL 32603 USA (e-mail: pengw@ufl.edu; lilidu@ufl.edu).

Digital Object Identifier 10.1109/TITS.2021.3135648

Many studies consider scaling down a large-scale CSP by partitioning riders into multiple smaller groups and exploring the carpooling solution for each group independently to address the scalability issue. However, simply partitioning based on geographical locations will compromise the system performance because it ignores possible carpooling routes that accommodate the riders from different groups. We want to address this dilemma by putting those riders with a high chance to share rides in one cluster and splitting riders without (or with few) carpooling chances into different clusters. To carry on this thought, we notice that riders have different opportunities to share rides due to the constraints relevant to their trips. It is critical to quantify their carpooling chances and conduct partition based upon that. In addition, from the system perspective, not all carpooling routes benefit system travel-time saving equally. We want to promote those carpooling routes, which help mitigate system service time better. These thoughts invoke our study to investigate quantitative approaches for evaluating the carpooling chances between riders thoroughly. Based on the carpooling chances between riders, we can form the desired carpooling clusters. Within each cluster, we explore the optimal carpooling options for the riders.

This study further recognized that clustering algorithms are usually unsupervised learning approaches. Thus the size of each cluster is not under control. Even though the cluster detection shrinks the size of the original CSP problem for sure, its solution can still contain a cluster with a relatively large number of riders. Accordingly, searching for the optimal carpooling solution within a carpooling cluster may still be a computation obstacle. Therefore, to further secure the computation efficiency, we transfer the combinatorial problem to a network flow problem so that we can explore feasible carpooling routes efficiently.

We summarize the mathematical methodology contribution of this study as follows. To mathematically implement the above ideas, we first contribute a mixed-integer program (CSP-MIP) to rigorously describe the CSP with pick-up and drop-off time window constraints. The suggested optimal carpooling routes should satisfy riders' trip time constraints while aligning with the interests of both TNCs and the society in saving system service time and fleet size. Thus, the optimal solution is expected to be highly accepted by all stakeholders. This CSP-MIP covers a general carpooling problem in literature but the mathematical model is missed in many relevant studies. Next, we develop the SCM approach, which decomposes the CSP-MIP and efficiently solves the sub-MIPs by integrating the following methods. To be noted, our approach provides the prescription solution from the supplier perspective, assuming a rider will accept the system assigned optimal carpooling route if the route can satisfy his/her pick-up and drop-off time window as well as the car capacity.

We first analyze the carpooling chance among riders and develop mathematical formulations to quantify the carpooling chance (C2) among the riders according to their itineraries, pick-up, drop-off time window constraints, and the potential system service time-saving. To the best of our knowledge, this research is barely investigated in the existing literature. Next, relying on the quantified C2 between riders, we construct a C2 graph and then use a community detection approach to find the desired carpooling communities/clusters, which ensure that the riders have higher C2 between riders within a community but not between riders in different communities. These mutually excluded carpooling communities facilitate decomposing a large-scale CSP-MIP over the entire network to several small sub-MIPs in local areas while minimizing the compromise of system optimality.

Next, we investigate the solution space of each sub-MIP and solve each of them independently using a network flow approach. Specifically, we analyze the feasible connections between two riders' origins and destinations, considering their pick-up and drop-off order, time window, and precedence requirements. With this knowledge, we transfer the combinatorial solution space of each sub-MIP to a graphic solution space only involving these feasible connections. We notice that each path starting with a rider's origin in the graph represents one feasible carpooling route if it satisfies the carpooling route constraints such as seat capacity. Accordingly, we design a carpooling tree generation algorithm (CTA) to efficiently search all feasible carpooling routes in a carpooling community. Three reasons guarantee the computation efficiency of the CTA: 1) the number of the trees is limited by the small size of the carpooling community; 2) the capacity of each CAV limits the depth of the carpooling tree; 3) the tree is efficiently searched on the graphic solution space. Once we obtain all feasible carpooling routes, a greedy algorithm is applied to optimally match carpooling routes and CAVs, aiming to serve all riders with the minimum service time and fleet size.

Our experiment results well justified the merits of the SCM approach and validated its applicability in reality. More exactly, compared to the geographical partition approach, the carpool-chance-based community detection is a better way to decompose the CSP-MIP while maintaining system performance. The CTA can identify a solution of a sub-MIP with a competitive quality as a Gurobi solution while taking much less computation time. For example, it took 0.26 seconds for the CTA to find a solution with a similar system performance as Gurobi found in 600 seconds. Moreover, our experiments demonstrate that the SCM outperforms several existing heuristic approaches developed in the literature to solve the CSP. For example, compared to the naïve ride-matching approach (NRM), the SCM reduces 65% of system service time and 56% of the number of vehicles used to fulfill the services in the system on average. The SCM took less than 1% of the computation time of the maximum cluster algorithm (MCA [13]) to identify a slightly worse solution in system performance than the solution of the MCA. Overall, the experiments demonstrate that the SCM dramatically improves computational efficiency while providing a solution with competitive quality.

The following structure presents the effort to conduct this research. Section II first reviews the most related research to demonstrate the research gaps in the literature. Then, we mathematically formulate the problem in Section III. Followed that, Section IV builds the mathematical model of the CSP and then develops the SCM approach. Section V conducts numerical experiments to validate the applicability and merits of our approach. The paper summarizes the research contributions and proposes future work in Section VI.

II. Literature Review

The CSP that interests this study can be considered as an online multiple-vehicles-multiple-riders ride-matching problem with time constraints. There are two mainstream approaches in the literature to solve such CSP: centralized and decentralized optimization. While centralized optimization techniques [7], [8] provide system optimal solutions, they often meet the difficulty of the scalability issue for this online application. On the other hand, decentralized optimization is of more interest since it generates solutions fast, but compromises the optimality. Accordingly, two types of decentralized optimization approaches have been developed to address the ride-matching problem: agent-based approach and partition-based approach. The brief review below recognizes these main efforts in the existing literature and highlights this study's unique contributions.

The agent-based approach solves dynamic ridesharing problems more efficiently than centralized optimization through autonomous agents optimizing their individual objectives based on the available local information acquired from the system. The study in [11] considers each driver as an independent agent, which communicates and serves nearby riders within a short communication range, using a heuristic approach to form the service routes. The study of [9] on the other hand introduces a ridesharing system considering each rider as an agent, which searches for the best potential driver every two minutes in the network. Both models meet the online application requirement, however, they have no attempt to optimize the travel time or travel cost at a system level. The study of [3] develops an agent-based model where drivers bid on their vicinity riders, and the riders decide if they accept the bid. The objective of drivers or riders is to maximize their revenue or minimize their travel time. Their approach does provide an optimal solution for a small group of riders, nonetheless, this heuristic approach is hard to extend to a large group of riders who want to and able to share rides, since it requires enumerating all possible carpool options for all the riders, and it's cumbersome to do so for a carpooling community with a large number of riders.

The approach developed by this study is closer to the partition-based approaches, which first separate the users into several smaller groups, then find the ride-match for each group instead of for the entire group of users. In this way, a large-size ride-matching problem can be decomposed into several smaller problems, resulting in better computation efficiency. Several pioneer studies first investigated the idea of partitioning the riders based on their geographic locations. For example, the study in [12] only allows riders and drivers to share a ride if their geographic pick-up locations are within two miles. This geo-partition based upon pick-up locations may lead to a long travel time delay due to a bad detour since the routes serving them to the destinations can be very different, even though their pick-up locations are close. The study of [13] proposed a greedy method, which sequentially calls a vehicle and assigns as many riders nearby until all riders are set. This approach may cause a long waiting time for the riders who are at the end of the assignment queue. A few recent studies [23]–[25] started the idea of clustering-aided approaches. They evaluate the shareability among riders, then cluster riders, and further reason ridesharing solutions on each cluster. However, the shareability in the studies of [23], [24] is evaluated according to the geographic distances between riders' pick-up locations, which does not explicitly capture potentially carpooling routes between them. The study of [25] identified the dissimilarity between two riders by measuring their temporal (time window difference) and spatial proximity (average detour time). But their effort does not consider pick-up and drop-off time windows like this study, which will bring in more realities, promote compliance rate, but raise extra modeling and computation complexity. We also noticed that these studies often use a predefined number of clusters, which is not realistic for the application in practice. State of the art shows we are still struggling with system optimality, scalability, and user compliance issues for efficiently solving a large-scale CSP.

From a more general view, this CSP problem is closely related to the Pick-up and Delivery Problem with Time Window (PDPTW) [28]–[30], [32] in literature. However, the PDPTW mainly focuses on planning the pickup and delivery routes for a vehicle fleet, considering a large capacity truck, relatively long lead time, and a small customer size (less than 100). The application is very different from the CSP, which involves a large-scale demand requiring quick online service responses for a small car fleet. Accordingly, the existing approaches developed for solving a PDPTW [28]–[30] demonstrate the computation inefficiency to be implemented online for solving the CSP. It calls for new solution approaches for the CSP considering its unique application features.

This study will address those research gaps mentioned above by developing a semi-centralized ride-matching approach aiming to balance system optimality and computation efficiency. It involves the mathematical formulation of the CPS, a new partition approach based on the quantitative carpooling chance measurement, and a network flow algorithm combined with a greedy match algorithm to search for optimal carpooling route solution. We introduce our approach in the following sections.

III. Problem Statement

This study seeks to help TNC explore a real-time optimal carpooling solution for riders requiring carpooling services, aiming to minimize total system travel time of vehicles (i.e., system service time) while satisfying individual riders' pick-up and drop-off time window requirements. It factors the benefits from both suppliers and demands, thus implies a potential high compliance rate for the carpooling services. In practice, passenger demands are changing in real time. We will run this carpooling solution minute by minute to service different batches of passengers.

Figure 1 illustrates an example for the CSP of interest. Specifically, it involves three CAVs respectively located at n_1, n_2, and n_3, and three riders $\{i, k, j\}$, each with its origin,

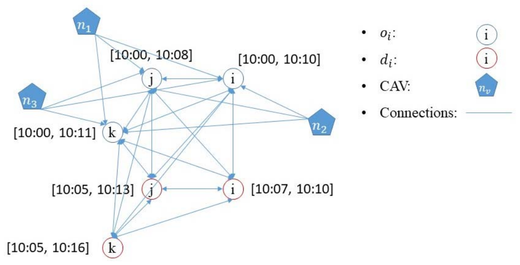


Fig. 1. Carpooling network.

destination, and pick-up and drop-off time windows. The arcs represent the road connections between the pick-up and drop-off spots in a city network. The carpool system needs to decide the optimal carpooling routes for CAVs so that they can serve all riders on the network while minimizing the overall system service time. Below provides the formal problem set-up so that we can develop mathematical approaches to address it.

We consider an online carpooling service system in a city network, which is modeled by a directed network $G(N, A)$ with the node set $N = O \cup D \cup \tilde{N}$ and the link set A. Here, O and D respectively represent the origins and destinations of riders, while $\tilde{N}$ represents the initial locations of CAVs. Accordingly, n is introduced as the index of a node on the network. Each link a is associated with known real-time travel time $\tau_a, a \in A$. Clearly, the CSP involves two sets of subjects: CAVs which provide carpooling services, and riders who are interested in the carpooling service. All such CAVs form a fleet denoted by $V = \{v\}$, and all riders form a set denoted by $I = \{i\}$. The riders are differentiated by their origins and destinations (ODs), pick-up time windows, and drop-off time windows. Here we assume that each carpool request only consists of one rider for model simplicity. Later, we will show that this assumption can be easily relaxed by a small modification in the mathematical model. By denoting the origin and destination of rider i by $o_i \in O$, and $d_i \in D, \forall i \in I$, we further define the pick-up and drop-off time windows of rider i by $[t^+(o_i), t^-(o_i)]$ and $[t^+(d_i), t^-(d_i)]$. Here, $t^+(o_i)$ and $t^-(o_i)$ respectively denote the service request time and the latest pick-up time, and $t^+(d_i)$ and $t^-(d_i)$ respectively represent the earliest and latest drop-off time. Each carpooling route $u \in U$ is defined as a path connecting a sequence of pick-up and drop-off locations while satisfying the time window constraints. Accordingly, there are at most four feasible carpooling routes between any two riders. For example, for two riders i and j, one of the possible carpooling routes is $u_{ij} = \{o_i \rightarrow o_j \rightarrow d_i \rightarrow d_j\}$. The actual path between any two locations on the carpooling route is the shortest path based on real-time traffic information to avoid traffic congestion. We assume that a rider will take the carpooling route if the pick-up and drop-off time windows are satisfied, and won't complain about the detour. Considering riders often avoid transferring, we do not consider this type of route in this study. We denote the travel time of a carpooling route as t_u. The travel time saving by a carpooling route u for the system serving each rider individually is denoted as Δ_u. Accordingly, the travel time saving of the route u_{ij} in the example above is $\Delta_{u_{ij}} = t_{o_i o_j} + t_{o_j d_i} + t_{d_i d_j} - t_{o_i d_i} - t_{o_j d_j}$ Last, tor each CAV $v \in V$, we assume a known initial location denoted by $n_v \in \tilde{N}$ (either idle at a depot or en route) and the capacity denoted by q^v. A CAV will serve a carpooling route involving multiple riders. It takes the travel time t_{uv} for CAV v to approach the start location of s carpooling route $u \in U$. We introduce our methodology to model and solve this CSP problem in the next section.

IV. Methodology

This study will build a mathematical program to describe the CSP problem rigorously and then develop SCM approach to find a local optimal solution efficiently by taking advantage of the unique features of the problem structure.

A. Mathematical Modeling

This study models the CSP problem defined in the problem statement section by a mixed-integer mathematical program shown by Eqs. (1)-(25), which seeks to find the optimal carpooling routes for the riders, each with a service time window requirement at the pick-up and drop-off locations, so that we can save the system service time with a small CAV fleet. To be noted, the system service time is the travel time of the CAV fleet to serve all the riders. The mathematical program will give us a concrete scientific base to develop efficient solution approaches. In addition, we will also use its solution solved by commercial software or other heuristic algorithms in literature as a benchmark to evaluate the performance of our solution approach. To do that, we first introduce the decision variables in the list below.

- $x^v_{n_1,n_2}, n_1, n_2 \in N, v \in V$: binary variables. $x^v_{n_1,n_2} = 1$ if CAV v travels along the arc from node n_1 to n_2, otherwise, $x^v_{n_1,n_2} = 0$.
- $y^v_{o_i}, v \in V, i \in I$: Integer variable, representing the occupancy of CAV v after picking up rider i.
- $\overleftarrow{t}^v_n, v \in V, n \in N$: Continuous variables representing the arriving time of CAV v at node n.
- $\psi^v_{o_i,o_j}, v \in V, i, j \in I$: Binary variable. $\psi^v_{o_i,o_j} = 1$, if the arriving time at o_i and $o_j$satisfies $\overleftarrow{t}^v_{o_j} \leq \overleftarrow{t}^v_{o_i}$; otherwise 0.
- $\psi^v_{o_i,d_j}, v \in V, i, j \in I$: Binary variables. $\psi^v_{o_i,d_j} = 1$, if the arriving time at d_j and o_i satisfies $\overleftarrow{t}^v_{d_j} \leq \overleftarrow{t}^v_{o_i}$; otherwise 0.
- $\omega^v_{(n_1,n_2),o_i}, \forall i \in I, \forall v \in V, n_1 \in \{O \backslash o_i\} \cup D, n_2 \in O \cup D, n_1 \neq n_2$: Binary variable. $\omega^v_{(n_1,n_2),o_i} = 1$ if link $(\mathrm{n}_1, \mathrm{n}_2)$ belongs the route of CAV v and CAV v visits node n_1 before o_i; otherwise $\omega^v_{(n_1,n_2),o_i} = 0$.

M is a very large number and ε is a very small number in the mathematical model. We next explain the objective function and the constraints in detail.

CSP-MIP:

$$min \sum\nolimits_{v \in V} \sum\nolimits_{n_1 \in O \cup D} \sum\nolimits_{n_2 \in O \cup D} \tau_{n_1 n_2} x^v_{n_1,n_2} + \sum\nolimits_{v \in V} \sum\nolimits_{n_2 \in O} \tau_{n_v n_2} x^v_{n_v,n_2} \quad (1)$$

Subject to

$$x^v_{n_1,n_2} = 0, \quad \forall v \in V, \quad \forall n_1 \in \tilde{N}, \quad \forall n_2 \in O \cup D, \quad n_1 \neq n_v \quad (2)$$

$$x^{\upsilon}_{n_1,n_2} = 0, \quad \forall \upsilon \in V, \quad \forall n_1 \in O \cup D \cup \tilde{N}, \quad \forall n_2 \in \tilde{N} \tag{3}$$

$$\sum\nolimits_{n \in O} x^{\upsilon}_{n_\upsilon,n} \le 1, \quad \forall \upsilon \in V \tag{4}$$

$$\sum\nolimits_{\upsilon \in V} \sum\nolimits_{n \in O \cup D \cup \tilde{N}} x^{\upsilon}_{n,o_i} = 1, \quad \forall i \in I, \quad n \ne o_i \tag{5}$$

$$\sum\nolimits_{n_1 \in O \cup D \cup n_\upsilon} x^{\upsilon}_{n_1,o_i} = \sum\nolimits_{n_2 \in O \cup D} x^{\upsilon}_{n_2,d_i}, \quad \forall i \in I, \quad \forall \upsilon \in V, \quad n_1 \ne o_i, \quad n_2 \ne d_i \tag{6}$$

$$\sum\nolimits_{n_1 \in \{O \backslash o_i\} \cup \{D \backslash d_i\} \cup \tilde{N}} x^{\upsilon}_{n_1,o_i} = \sum\nolimits_{n_2 \in \{O \backslash o_i\} \cup D} x^{\upsilon}_{o_i,n_2}, \quad \forall i \in I, \quad \forall \upsilon \in V \tag{7}$$

$$\sum\nolimits_{n_1 \in O \cup D} x^{\upsilon}_{n_1,d_i} \ge \sum\nolimits_{n_2 \in O \cup D} x^{\upsilon}_{d_i,n_2}, \quad \forall i \in I, \quad \forall \upsilon \in V, \quad n_1 \ne d_i, \quad n_2 \ne d_i \tag{8}$$

$$\vec{t}^{\,\upsilon}_n \ge 0, \quad n \in O \cup D \cup \tilde{N} \tag{9}$$

$$\overleftarrow{t}^{\,\upsilon}_{n_2} - \vec{t}^{\,\upsilon}_{n_1} - \tau_{n_1 n_2} \ge M\left[x^{\upsilon}_{n_1,n_2} - 1\right], \quad \forall \upsilon \in V, \quad \forall n_1 \in O \cup D \cup \tilde{N}, \quad \forall n_2 \in O \cup D, \quad n_1 \ne n_2 \tag{10}$$

$$\vec{t}^{\,\upsilon}_{n_1} - \overleftarrow{t}^{\,\upsilon}_{n_1} \ge \varepsilon \sum_{n_2 \in O \cup D \backslash n_1} x^{\upsilon}_{n_1,n_2}, \tag{11}$$
$$\forall \upsilon \in V, \quad \forall n_1 \in O \cup D \cup \tilde{N}$$

$$\overleftarrow{t}^{\,\upsilon}_{o_i} \le \overleftarrow{t}^{\,\upsilon}_{d_i}, \quad \forall i \in I, \quad \forall \upsilon \in V \tag{12}$$

$$\overleftarrow{t}^{\,\upsilon}_n \le t^-(n) - t^+(n), \quad \forall n \in O \cup D \tag{13}$$

$$\overleftarrow{t}^{\,\upsilon}_{o_i} - \overleftarrow{t}^{\,\upsilon}_{o_j} \le M\psi^{\upsilon}_{o_i,o_j}, \quad \forall i \in I, \quad \forall j \in I, \quad i \ne j \tag{14}$$

$$\overleftarrow{t}^{\,\upsilon}_{o_i} - \overleftarrow{t}^{\,\upsilon}_{o_j} \ge M\left(\psi^{\upsilon}_{o_i,o_j} - 1\right), \quad \forall i \in I, \quad \forall j \in I, \quad i \ne j \tag{15}$$

$$\overleftarrow{t}^{\,\upsilon}_{o_i} - \overleftarrow{t}^{\,\upsilon}_{d_j} \le M\psi^{\upsilon}_{o_i,d_j}, \quad \forall i \in I, \quad \forall j \in I, \quad i \ne j \tag{16}$$

$$\overleftarrow{t}^{\,\upsilon}_{o_i} - \overleftarrow{t}^{\,\upsilon}_{d_j} \ge M\left(\psi^{\upsilon}_{o_i,d_j} - 1\right), \quad \forall i \in I, \quad \forall j \in I, \quad i \ne j \tag{17}$$

$$\omega^{\upsilon}_{n_1,n_2,o_i} \le x^{\upsilon}_{n_1,n_2}, \quad \forall i \in I, \quad \forall \upsilon \in V \quad \forall n_1 \in \{O \backslash o_i\} \cup D, \quad n_2 \in O \cup D, \quad n_1 \ne n_2 \tag{18}$$

$$\omega^{\upsilon}_{n_1,n_2,o_i} \le \psi^{\upsilon}_{o_i,n_1}, \quad \forall i \in I, \quad \forall \upsilon \in V, \quad n_1 \in \{O \backslash o_i\} \cup D, \quad n_2 \in O \cup D, \quad n_1 \ne n_2 \tag{19}$$

$$\omega^{\upsilon}_{n_1,n_2,o_i} \ge x^{\upsilon}_{n_1,n_2} + \psi^{\upsilon}_{o_i,n_1} - 1, \quad \forall i \in I, \quad \forall \upsilon \in V, \tag{20}$$
$$n_1 \in \{O \backslash o_i\} \cup D, \quad n_2 \in O \cup D, \quad n_1 \ne n_2$$

$$\sum\nolimits_{n_1 \in \{O \backslash o_i\}} \sum\nolimits_{n_2 \in O \cup D} \omega^{\upsilon}_{n_1,n_2,o_i} + \sum\nolimits_{n_5 \in O \cup D \cup \tilde{N} \backslash o_i} x^{\upsilon}_{n_5,o_i} - \sum\nolimits_{n_3 \in D} \sum\nolimits_{n_4 \in O \cup D} \omega^{\upsilon}_{n_3,n_4,o_i} = y^{\upsilon}_{o_i}, \quad \forall i \in I, \quad \forall \upsilon \in V, \quad n_1 \ne n_2, \quad n_3 \ne n_4 \tag{21}$$

$$y^{\upsilon}_{o_i} \le q^{\upsilon}, \quad \forall \upsilon \in V, \quad \forall i \in I \tag{22}$$

$$x^{\upsilon}_{n_1,n_2} = \{0, 1\}, \quad \forall n_1 \in O \cup D \cup \tilde{N}, \quad \forall n_2 \in O \cup D, \quad n_1 \ne n_2 \tag{23}$$

$$\psi^{\upsilon}_{o_i,n} = \{0, 1\}, \quad \forall i \in I, \quad \forall n \in O \cup D, \ \textit{and} \ n \ne o_i \tag{24}$$

$$\omega^{\upsilon}_{n_1,n_2,o_i} = \{0, 1\}, \quad \forall n_1 \in O \cup D, \quad \forall n_2 \in O \cup D, \quad n_1 \ne n_2, \quad \forall i \in I \tag{25}$$

The objective function seeks to minimize the system service time of all active vehicles serving all the riders, which matches the interests of all stakeholders. To be noted, the objective function implicitly penalizes the number of active vehicles so that the optimal solution determines a small fleet size to fulfill the minimum system service time presented in the objective function. The small fleet will help mitigate the traffic congestion overhead introduced by the ridesharing vehicles. Eq. (2) limits that the route of CAV υ can only start at its pre-defined initial location. Eq. (3) ensures that a carpooling route won't go back to CAVs' initial locations during a trip. Eq. (4) limits that an active CAV υ can only pick one rider at a time. Eq. (5)-(8) ensures that a rider's origin and destination must be visited and are only visited by one CAV once. More exactly, Eq (5) requires that the origin of a rider can only be visited once by only one CAV. Eq. (6) further ensures that if a CAV υ picks up rider i at the origin o_i, it must drop rider i off at the destination d_i. Eq. (7) presents the flow balance of CAV υ at an origin node. Namely, if CAV υ arrives at rider i's origin o_i should also leave the origin o_i. Eq. (8) denotes that CAV υ will stop at destination d_i if it is the end of the carpooling route. Eq. (9) requires the arriving time at each node is non-negative.

Eqs. (10-12) together guarantees the sequential visiting order of the carpooling route of CAV υ. Namely, Eq. (10) ensures the temporal feasibility for a CAV υ that sequentially visits two nodes (i.e., the arriving time at the latter node is no earlier than the leaving time from the precedent node plus the travel time between these two nodes). Eq. (12) requires that CAV υ should arrive at a node before it leaves from this node. Eq. (12) ensures that rider i is first picked up at its origin and then dropped off at its destination (i.e., a carpooling route of CAV υ visits rider i's origin first and then its destination.)

Eq. (13) presents the time windows of riders at their origins and destinations. Eqs. (14)-(15) identifies whether the pick-up time of rider j is no later than that of rider i. If yes, $\psi^{\upsilon}_{o_i,o_j}$ takes value 1; otherwise, it equals 0. Similarly, Eqs. (16)-(17) judge whether CAV υ drops off rider j no later than the time it picks up rider i. If yes, $\psi^{\upsilon}_{o_i,d_j}$ takes value 1; otherwise, it equals 0. Eqs. (18)-(21) detect the visiting order between n_1 and o_i. Namely, if node n_1 is visited before o_i by CAV υ, given n_1 is visited (i.e., $x^{\upsilon}_{n_1,n_2} = 1$) and the arriving time of n_1 is no later than the arriving at o_i (i.e., $\psi^{\upsilon}_{o_i,n_1} = 1$), then $\omega^{\upsilon}_{n_1,n_2,o_i} = 1$; otherwise, $\omega^{\upsilon}_{n_1,n_2,o_i} = 0$. Using the values of $\omega^{\upsilon}_{n_1,n_2,o_i}$, Eq. (21) is able to count the number of riders that have been picked up/dropped off by CAV υ when it arrives at o_i. Eq. (22) guarantees that a CAV is not overloaded. Eq. (23)-(25) claims that $x^{\upsilon}_{n_1,n_2}$, $\psi^{\upsilon}_{o_i,n}$, and $\omega^{\upsilon}_{n_1,n_2,o_i}$ are binary variables. Note that the CSP-MIP will determine a small fleet size to fulfill the minimum system travel time to serve all the riders.

Note that the above model can be extended to solve the CSP, in which each carpool request involves multiple riders. Mainly, we will consider $\forall i \in I$ is a carpool request involving q_i number of riders with the same origin, destination, and time requirements (i.e., $q_{o_i} = q_{d_i} = q_i$). Accordingly, Eq. (21) above can be simply modified to Eq. IV-B for addressing the case, and the rest of model remains the same. The mathematical model is a mixed-integer program, which is NP-hard [17], [18]. The complexity of the problem increases dramatically as the number of riders involved increases. Thus, it usually cannot be efficiently solved when the problem sale is large. Our experiments show that the Gurobi solver takes 508 seconds to solve a case with 12 riders in a small network.

$$y_{o_i}^v = \sum_{n_1 \in \{O \backslash o_i\}} \sum_{n_2 \in O \cup D} q_{n_1} \omega_{n_1,n_2,o_i}^v + q_i \sum_{n_5 \in O \cup D \cup \tilde{N} \backslash o_i} x_{n_5,o_i}^v - \sum_{n_3 \in D} \sum_{n_4 \in O \cup D} q_{n_3} \omega_{n_3,n_4,o_i}^v, \quad \forall i \in I, \quad \forall v \in V, \quad n_1 \neq n_2, \quad n_3 \neq n_4 \tag{26}$$

Clearly, the standard solver cannot satisfy the online application, aiming to respond to online carpooling service requests. We address the computation difficulty in the next section.

B. Solution Approach

This study develops a semi-centralized ride-matching (i.e., SCM) approach to efficiently solve a large-scale CSP-MIP. It combines the decomposition and network flow algorithms to explore a local optimal carpooling solution while balancing the solution optimality and computation efficiency. Briefly, the SCM approach first analyzes the carpooling chances among riders and then develops a mathematical formulation to quantify them. Built upon that, we strategically partitioning the riders into carpooling communities by community detection approach. Equally, we decompose a large-scale master CSP-MIP involving many riders in a network to multiple small sub-MIPs, each with fewer riders in a community, seeking to scale down the CSP-MIP with a minor compromise of the system optimality. Next, we develop a network flow algorithm to search for a local optimal solution for each sub-MIP quickly. The main idea is to present the feasible solutions of the sub-MIP in a graphic solution space, develop a tree generation algorithm to enumerate all feasible carpooling routes efficiently, and then greedily match those routes with CAVs with the objective to minimize total service time using a small fleet. We introduce the technical detail for these methods in the following sections.

1) Carpooling Chance Analysis: To efficiently decompose the CSP-MIP without compromising the system optimality, ideally, we want to partition the riders in a strategic way so that riders in different groups have a much lower chance to carpool (i.e., not coupled in the constraint sets of the CSP-MIP). This thought invokes us to analyze the carpooling chance between any pair of riders in depth. Built upon the understanding, we want to quantify the carpooling chance and then form carpooling groups within which the riders have a high carpooling chance and between which the riders have a low carpooling chance. More exactly, it is noticed that not all riders have equal carpooling chances. Figure 2 shows an example. If the current time is 10:00, and travel time between o_i and o_j is 20 minutes, then it's impossible for rider i and j to carpool together. However, if $t_{o_io_j} = 5, t_{o_jd_i} = 5,$ and $t_{d_id_j} = 5$, then rider i and j have a good chance to do carpool since there exist at least a route $\{o_i \rightarrow o_j \rightarrow d_i \rightarrow d_j\}$ that serve both their origins and destination without violating their time windows. State of the art shows that we still lack a quantitative approach to measure this carpool chance.

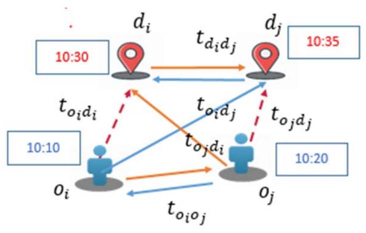


Fig. 2. Carpooling chance between i and j.

In view of the above gap, this study seeks to develop a novel formulation that helps us evaluate the chance that two riders can do carpool, considering several factors such as the location of their ODs, the trip plan, and so on. To do that, we use the following rules to identify an acceptable carpooling route from the perspectives of the riders and the system, respectively. First, from the riders' perspective, two riders will potentially carpool if there is at least one carpooling route, which satisfies the pick-up and drop-off time windows of both riders; otherwise, they will not. Taking riders i and j in Figure 2 as an example, the carpooling route $u_{ij} = \{o_i \rightarrow o_j \rightarrow d_i \rightarrow d_j\}$ is applicable if the travel time from o_i to o_j ($t_{o_io_j}$) is within the pick-up window of rider j (e.g., $t_{o_io_j} \leq (t^-(o_j) - t^+(o_j))$). Similarly, this route should also not violate the drop-off time window of rider i and rider j (e.g., $t_{o_io_j} + t_{o_jd_i} \leq t^-(d_i) - t^+(d_i)$, and $t_{o_io_j} + t_{o_jd_j} + t_{d_jd_i} + t_{d_id_j} \leq t^-(d_j) - t^+(d_j)$). Clearly, a looser pick-up or drop-off time window potentially leads to more acceptable carpool options from the rider's perspective. Let $\varpi_{u_{ij}}$ be the actual waiting time of carpooling route u_{ij}, and $\hat{\varpi}_{u_{ij}}$ is the length of the pick-up time window of the second rider on the carpooling route u_{ij}. We further conclude that a carpooling route between two riders involves a long waiting time ($\hat{\varpi}_{u_{ij}^k} - \varpi_{u_{ij}^k}$) buffer leads to a strong carpooling chance between riders. We next consider the rule from the system perspective. Specifically, a carpooling service is worth promoting if the carpooling route saves travel time compared to the total travel time needed to pick up and drop off each rider individually, i.e., $\Delta_{u_{ij}} = t_{o_io_j} + t_{o_jd_j} + t_{d_jd_i} - t_{o_id_i} - t_{o_jd_j} \geq 0$ should hold for the example in Figure 2. Clearly, a high value of the system service time saving $\Delta_{u_{ij}}$ indicates a more favorable carpooling service from the system point of view.

Using the above two rules, we can identify a feasible carpooling route set between each pair of riders, U_{ij}, $\forall i \neq j \in I$ given there are at most four possible carpooling routes between them. And then, we quantify the carpooling chance

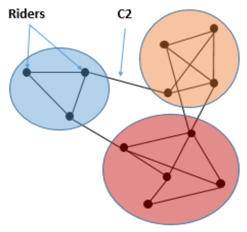


Fig. 3. Carpooling chance (C2) graph.

between any two riders i and j by Eq. (27) below.

$$c_{ij} = \sum_{U_{ij}} \left[\underbrace{\frac{\Delta u_{ij}^k}{t_{o_i d_i}} + \frac{\Delta u_{ij}^k}{t_{o_j d_j}}}_{\text{travel time saving}} \right] \left[\underbrace{1 - \frac{\varpi_{u_{ij}^k}}{\widehat{\varpi}_{u_{ij}^k}}}_{\text{waiting time buffer}} \right] \quad \forall i \neq j,\ i,\ j \in I, \tag{27}$$

where c_{ij} denotes the quantitative carpool chance between two riders i *and* j; $u_{ij}^k \in U_{ij}$ denote the k-th carpooling route between i and j, Δu_{ij}^k is the system service time saving resulting from the carpool. Specifically, Eq. (27) considers three factors as follows. First of all, the system promotes a carpooling route if it leads to the saving of system service time. Mathematically, it is measured by the saving of service time ratio in the first item of Eq. (27). Next, two riders will have a high chance to accept a carpooling route if the carpooling routes lead to a short waiting time for the second rider. This factor is captured by the waiting time ratio formulated by the second item in Eq. (27). Last, we consider two riders have higher chances to do carpool if there are more feasible carpooling routes between them. Therefore, Eq. (27) quantifies the value of carpool chance between two riders by summing up the carpooling possibilities over all feasible carpooling routes.

Built upon the quantified carpooling chance between two riders, we construct the carpooling chance graph $\Gamma(I,\ L)$ to present the potential carpools among riders. The node set I denotes all the riders and the edge set L involves all feasible carpool relationships between two riders. The weight of each link between a node pair i and j is the carpooling chance, i.e., c_{ij}, $\forall i, j \in I$ and link $(i, j) \in L$. Figure 3 shows an example.

2) Carpool Communities on the Carpooling Chance Graph: The carpooling chance graph presents very useful information about the carpool chances among riders. It will facilitate us to develop a proper approach to decompose the CSP-MIP and improve the computation efficiency. Specifically, we can see that some riders are not connected in the graph. It means that they are not able to do carpooling. But other riders form local cliques in the graph, which show great carpool chances among them. These observations motivate us to partition the riders according to the cliques in the graph. In the other words, we can decompose a large-scale CSP into multiple small-scale CSP according to the cliques formed in the carpooling chance graph. We expect that riders present high (low) carpooling chances within (between) the cliques so that the decomposition won't significantly compromise the system optimality. Note that this is different from the existing approaches which partition riders according to their geographic locations of the pick-up or drop-off locations. The carpool chances are not well incorporated by those partition approaches.

To find those carpool cliques in the carpooling chance graph, we use a graph clustering algorithm. Among various algorithms developed in the graph clustering and community detection literature, we choose to use the hierarchical agglomeration algorithm developed by [5] since its computational complexity is linear for many real-world networks. For completeness, we briefly introduce this algorithm as follows. This community detection method is based on a measure named modularity, which measures the density of the connections within communities and between communities [6]. The formulation of the modularity is given in the equation below.

$$Z = \frac{1}{2m} \sum_{ij} \left[c_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(\pi_i, \pi_j\right) \tag{28}$$

where π_i denote the community that node i belongs to; $\delta\left(\pi_i, \pi_j\right) = 1$ if node i and j belong to the same community (e.g. $\pi_i = \pi_j$); otherwise, $\delta\left(\pi_i, \pi_j\right) = 0$; k_i is the sum of the weights (i.e., the values of carpooling chances) of all edges going out from node i, and $m = \frac{1}{2}\sum_{ij} c_{ij}$ is the sum of weights of all edges in the graph. The fraction $\frac{k_i k_j}{2m}$ is the expected weight of edges between nodes i and j where the maximum number (i.e., m) of edges are assigned between two nodes. Thus, a larger value of Z indicates stronger weights among riders within the communities. Following the terminology of the clustering algorithm, this study calls "carpool clique" as "carpool community" in this study.

The community detection starts with where every rider forms a community. Each time the algorithm joins two communities that result in the biggest increase of modularity. The modularity measures how strong the community structure is over a random assignment of edges. Accordingly, this algorithm aims to find a clustering solution that maximizes the modularity Z where dense connections are within communities and few there are between communities. In our case, it means riders have a stronger carpooling chance within the same community than the riders from other communities.

To be applied on the carpooling chance graph, the community detection starts with where every rider forms a community. Each time, it joins two communities that result in the most significant increase of modularity. It will stop until none of the joint operations improve the modularity score. Note that the community detection algorithm is an unsupervised learning method, which can neither ensure the number of the communities nor the size of each community. However, large communities are not preferred for this study since it will result in a relatively high computation load to search carpooling routes. Accordingly, we attempt to limit the size of a community by an upper bound with the insights as follows. The capacity of a CAV is relatively small so that potentially each rider is only able to share the ride with a limited number of riders. The marginal effect of improving carpooling solution performance by involving more riders in a community is diminishing. Namely, we conjecture that if the capacity of the CAV is small, such as 4, the candidate carpooling routes for a single rider won't increase infinitely as the size of the community expands, such as from 10 to 50 or even more riders. This is because a rider most likely won't have C2 with extra riders due to its trip origin and destination and service time window request. Our numerical experiments validate this idea.

3) Forming Graphic Solution Space for the Feasible Carpooling Routes in a Community: Even though forming carpooling community helps reduce the size of the CSP-MIP, existing literature [19] and also our numerical experiments demonstrated that standard solution approaches in commercial software such as the Gurobi still cannot meet the computation efficiency requirement for this online application, even when each carpooling community only involves 20 riders. The main computation obstacle is the huge combinatorial solution space. To address this difficulty, we come to an idea that prunes the network, involving all physical roads between riders' origins and destinations (an example shown in Figure 1) to a graphic space, including possible carpooling connections between riders' origins and destinations according to the constraint set of the CSP-MIP. We will then search for the optimal carpool routes based on this graphic solution space. More exactly, to present the combinatorial solutions defined in Eqs. (5)-(22), this study notices that a feasible carpooling route in $G(N, A)$ can be considered as a path formed by the temporal-spatial possible connections between the riders' origins and destinations (i.e., feasible solutions for $x^{v}_{n_1,n_2}, n_1, n_2 \in N, v \in V$ in the CSP-MIP). Along with this thought, this study considers that all feasible connections among riders in a community form a graph, which further form feasible carpooling routes. More importantly, this graphic solution space should rule out many infeasible carpooling routes to expedite the optimal carpooling routes searching. This section introduces our approach to construct this graph. The next section will develop a network flow algorithm built up this graph to search optimal carpooling routes for riders within each community.

It is noticed that a carpooling route can be formed by four types of connections between the origins and destinations of two riders: origin-to-origin ($O \rightarrow O$), destination-to-destination ($D \rightarrow D$), origin-to-destination ($O \rightarrow D$), and destination-to-origin ($D \rightarrow O$). Each of them corresponds to a special pick-up and/or drop-off sequence involved in a carpooling route. We will analyze the conditions to ensure the feasibility of these four types of connections considering riders' service requests. Built upon that, this section forms a graphic solution space consisted of feasible carpooling routes between two riders in a community. Note that without loss of generality, we develop the feasibility conditions for those connections by considering any two riders such as $\forall i \neq j, i$ and $j \in I$.

We first analyze the feasibility of an $O \rightarrow O$ connection between two riders. If an $O \rightarrow O$ connect is feasible, it indicates that a carpooling route will pick up rider i, and then rider j at their origins without violating the time window constraints of both riders. More exactly, the following constraints have to be satisfied. 1) The carpooling route does not violate the pick-up time window of rider j: $t_{o_i o_j} \in [t^{+}(o_j), t^{-}(o_j)]$. 2) The detour to pick up the rider j won't violate the drop-off time window of the rider i at its destination: $t_{o_i o_j} + t_{o_j d_i} \in [t^{+}(d_i), t^{-}(d_i)]$. See the example in Figure 5 for both conditions. This consideration is consistent with Eq. (13), which represents the time window constraints of riders' origins and destinations in the MIP. With this in mind, we construct a directed graph G^{O} (e.g., Figure 4 (a)) which involves all feasible $O \rightarrow O$ connections among all riders. Accordingly, each node in G^{O} represents the origin of a rider, and a directed edge from a node o_i to node o_j only exists if it satisfies the constraints of this $O \rightarrow O$ connection and the weight w_{ij} on the directed edge from o_i to o_j is the travel time from o_i to o_j, $\forall i$ and $j \in I$.

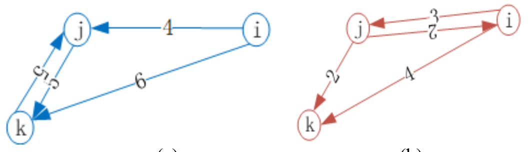


Fig. 4. (a) Origin graph only involving the feasible origin-to-origin connections; (b) Destination graph only involving the feasible destination-to-destination connections.

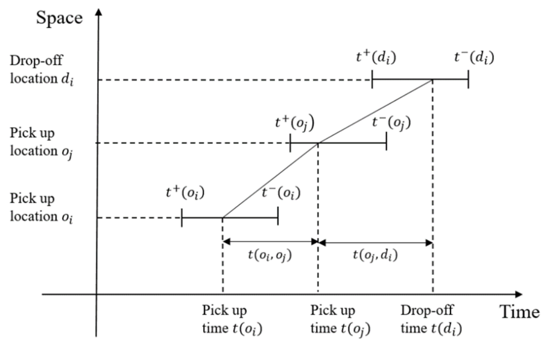


Fig. 5. Origin feasibility between rider i and j.

Next, we discuss the feasibility of a $D \rightarrow D$ connection between riders. Similar to a feasible $O \rightarrow O$ connection, a feasible $D \rightarrow D$ connection indicates that a carpooling route will drop off rider i then j without violating their time constraints. If a carpooling route drops off rider i then j, there are only these two possible carpooling routes between rider i and j: $\{o_i \rightarrow o_j \rightarrow d_i \rightarrow d_j\}$ or $\{o_j \rightarrow o_i \rightarrow d_i \rightarrow d_j\}$, both of which should satisfy rider j's drop off time window: $t_{o_i o_j} + t_{o_j, d_i} + t_{d_i d_j} \in [t^{+}(d_j), t^{-}(d_j)]$ or $t_{o_j o_i} + t_{o_i, d_i} + t_{d_i d_j} \in [t^{+}(d_j), t^{-}(d_j)]$. With these considerations, a feasible $D \rightarrow D$ connection will satisfy Eq. (13), which ensures feasible visiting order between both riders' origins and destinations, factoring their time constraints in the MIP. Following this thought, we construct a directed graph G^{D} (e.g., Figure 4 (b)) involving all feasible $D \rightarrow D$ connections. Each node in G^{D} represents the destination of a rider, and a directed edge from d_i to d_j only exist if they meet the feasibility condition of a $D \rightarrow D$ connection. Accordingly, the weight w_{ij} of the directed edge from d_i to d_j is the travel time from node d_i to node d_j.

It is noticed that riders' origins and destinations may interweave in a carpooling route. Accordingly, two more types of connections, i.e., $O \rightarrow D$ and $D \rightarrow O$, exist in a feasible

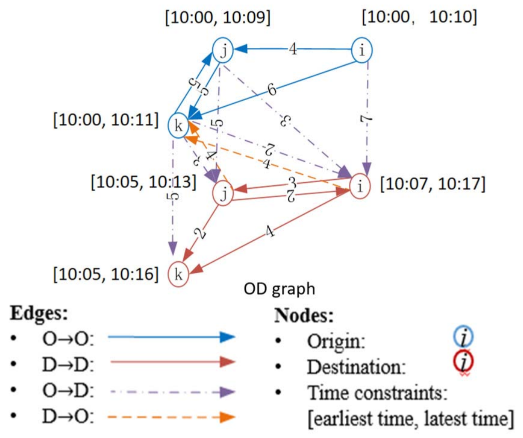


Fig. 6. Graphic solution space involving all feasible carpooling routes.

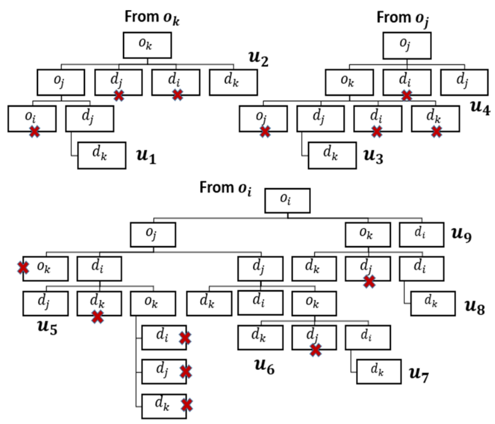


Fig. 7. Carpooling tree generation algorithm (CTA).

carpooling route. We further analyze the feasibility of these two types of connections so that we can combine the origin graph and the destination graph into a directed graph, which involves all feasible carpooling routes.

Regarding the feasibility of the connections of $O \rightarrow D$, we noticed that the rider's destination is always accessible from his own origin (e.g., $o_i \rightarrow d_i$). However, if rider j's destination is visited from another rider i's origin (e.g., by a connection$o_i \rightarrow d_j$), then the carpooling route that consists of these two riders will follow this visiting sequence $o_j \rightarrow o_i \rightarrow d_j \rightarrow d_i$. Namely, it has to visit rider j's origin o_j before rider i's origin o_i; moreover, the carpooling route can only visit rider i's destination d_i after it visits rider j's destination d_j. In addition, rider j' drop-off time windows have to be satisfied, i.e., $t_{o_j o_i} + t_{o_i d_j} \in [t^+(d_j), t^-(d_j)]$.

Following these two rules, we can build all feasible connections from the Origin graph to the Designation graph (i.e., $G^O \rightarrow G^D$).

Regarding the feasibility of the connections $D \rightarrow O$, this study is aware that we can't have a connector from a rider's destination to his origin (e.g., $d_i \rightarrow o_i$). Next, if the connection is from a rider i's destination to another rider j's origin (e.g. $d_i \rightarrow o_j$), rider i's destination is only visited after i's origin, the pick-up time window of o_j has to be satisfied, i.e., $t_{o_i,d_i} + t_{d_i,o_j} \in [t^+(o_j), t^-(o_j)]$. The connections from the destination group to the origin graph have to satisfy the above two rules (e.g., $G^D \rightarrow G^O$). Note that the feasible connections of $O \rightarrow D$ and $D \rightarrow O$ also satisfy constraints (13) in the MIP model, which generally ensure feasible visiting order between riders' origins and destinations factoring their time window limits.

Based on the above analysis for the four types of connections involved in a feasible carpooling route, this study constructs the directed graph by integrating the two graphs G^O and G^D. See an example shown in Figure 6. We denote this graph by $G(\mathcal{N}, E)$, in which $\mathcal{N} \subset N$ denotes the node set involving all riders' origins and destinations; it is a subset of nodes involved in the city network; E denotes the set of feasible connections which can be used to form carpooling routes; each connection is associated with a weight measured by the travel time. We name this graph as a graphic solution space for the carpooling route solutions. This is because it includes all feasible carpooling routes in a community. Please note that the connections in $G(\mathcal{N}, E)$ factors spatial, temporal feasibility as well as visiting order between riders' origins and destinations. Therefore, this graphic solution space greatly shrinks the size of the carpooling network by removing the unfeasible connections. Our next task is to find the best carpooling routes for riders in each community built upon this graphic solution space, aiming to minimize their detours and save the service time for the system as much as possible (i.e., saving the system service time).

4) Explore Optimal Carpooling Routes: Built upon the graphic solution space constructed for each carpooling community, we develop a carpooling tree generation algorithm (CTA) to explore all feasible carpooling routes in each community and further match carpooling routes with the CAVs by a greedy algorithm to minimize the system service time with a small CAV fleet size. We first discuss the main idea of the CTA algorithm. The CTA algorithm starts from the origin node of a random rider and expands on the graphic solution space to explore all feasible carpooling routes starting from it. Each complete path on the graphic solution space is a feasible carpooling route, and all carpooling routes starting from the same root form a carpooling route tree. The CTA is able to efficiently enumerate all the carpooling route trees in a community. Specifically, the CTA algorithm involves two main procedures.

i. Branching: this procedure starts from an origin in the graphic solution space of each community as the root node (such as o_i in Figure 7 and branches out (i.e., extends) the carpooling routes along with the neighborhood nodes in the graphic solution space until visiting all feasible carpooling routes (denoted by $\forall u \in U$) in a community.
ii. Cutting: Along with the branching procedure, the CTA cuts the infeasible branches (stops growing the route) along which a route violates the visiting sequence, vehicle capacity, or time constraints. (We explain the cutting conditions soon).

In order to traverse all feasible carpooling routes starting from one origin, the CTA involves two types of moves as follows. The two moves also reinforce the need to construct the graphic solution space since it greatly avoids infeasible exploration during the tree generation by removing the infeasible connections among riders' origins and destinations.

i. Forward-move: at each node, the CTA moves one node deeper along the current branch by exploring an unvisited neighbor node and then updates the travel/service time of the current branch/route.
ii. Backward-move: the algorithm backtracks one node along the current branch to explore new branches after it finishes a complete and feasible carpooling route, or the current branch is cut.

It is noticed that not every path between two nodes is potentially a feasible carpooling route, even though each connection is feasible. The primary goal of the algorithm is to find all feasible carpooling routes quickly. We then describe how to cut branches to avoid infeasible exploration. To do that, we first introduce extra notations as follows. Let $\hat{u}$ be the active route and $\hat{s}$ denotes the active node that the algorithm just visits. We use $t_{\hat{s}}$ to represent the arriving time from the origin node that the searching starts with to the active branch $\hat{s}$ on the active route $\hat{u}$, and $[t^{+}(\hat{s}), t^{-}(\hat{s})]$ is the arriving time constraints of node $\hat{s}$. $q_{\hat{u}}$ denotes the number of riders on the route $\hat{u}$, and q^{v} is the capacity of CAV.

Whenever the CTA branches and add a new node to the current carpooling route, it updates the travel time and checks the feasibility conditions by examining the following three conditions. 1) The riders at this node should not exceed the capacity of CAV, i.e., $q_{\hat{u}} < q^{v}$ (this corresponds to the CAV's capacity constraints Eq. (14)-(22) in the MIP). 2) The travel time of the route up to this new node should not violate the pick-up and drop-off time constraints of this node as shown in the time window constraint in Eq. (13) of the MIP, i.e., $t_{\hat{s}} \in [t^{+}(\hat{s}), t^{-}(\hat{s})]$ is required. 3) Visiting this node should not violate the precedence of riders' origins and destinations, i.e, if $\hat{s} = d_i$, then $o_i \in \hat{u}$ has to be true. Since the travel time is updated when a new node is added to the current route, this consideration echoes the visiting sequence conditions in Eqs. (9)-(12) of the MIP. Once an active node $\hat{s}$ violates the feasibility conditions, this active branch is cut and further exploration is stopped, and a backward move is carried to explore other new branches.

The CTA only keeps the complete routes, which involves both the origin and the destination of every rider. In addition, the forward-move only visits the unvisited neighbors of the current visiting node. Thus, it guarantees that every rider's origin and destination in a route is visited and only visited once. This ensures the constraints Eq. (5)-(8) in the MIP are satisfied.

Figure 7 shows an example of these exploration procedures. Starting from the root o_i and also following the connections in the graphic solution space, the algorithm finds a feasible carpooling route serving all three riders, namely, $(o_i \rightarrow o_j \rightarrow d_j \rightarrow o_k \rightarrow d_i \rightarrow d_k)$, while the infeasible branches are fathomed such as $(o_i \rightarrow o_k \rightarrow d_j)$ due to the violation of the precedence of riders' origin and destination.

Algorithm 1 Carpooling Tree Generation Algorithm for Exploring Feasible Carpooling Routes in a Community

```
Step 0: start from a node h on origin layer $G^O$, $\hat{s} = h$, $\hat{s} \rightarrow \hat{u}$, and $\hat{u}^1 = \hat{s}$;
Step 1: Forward-move:
  1) If $\hat{u}$ is a complete feasible carpooling route, $\hat{u} \rightarrow U$, go to step 2 backward search;
  2) Check if $\hat{s}^{\odot} = \hat{s}^{\otimes}$, if so, then go to Step 2 backward search;
  3) If not, visit an unexplored neighbor node $\hat{l}$ of active node $\hat{s}$, $\hat{l} \in \{\hat{s}^{\odot} \backslash \hat{s}^{\otimes}\}$, record the current travel time of route $\hat{u}$ at $\hat{l}$;
  4) Check the visit feasibility of $\hat{l}$,
     I. if $\hat{l}$ is not feasible to visit, then $\hat{l} \rightarrow \hat{u}^{1^{\otimes}}$ go to Step 1.2;
     II. if $\hat{l}$ is feasible to visit, then $\hat{s} = \hat{l}$ and $\hat{s} \rightarrow \hat{u}$ go to Step 1.1;
Step 2: Backward-move: $\hat{s} \rightarrow \hat{u}^{2^{\otimes}}$, and remove $\hat{s}$ from $\hat{u}$, $\hat{s} = \hat{u}^1$, and carry Step 1 forward search.
Step 3: End if $\forall s \in \Psi, s^{\odot} = s^{\otimes}$, and $\hat{s} = h$
```

This CTA algorithm will independently explore the carpooling tree starting from each individual origin in graphic solution space for each community. Each exploration is terminated when a root origin node becomes an active node again through the backward move. It means that the CTA has explored every neighbor node of this carpooling route tree. The pseudo-code of this algorithm is provided in Algorithm 1, in which $\hat{u}^m$ denotes the m-th visited node from last in the active branch $\hat{u}$. If a neighbor node $\hat{l}$ is infeasible to visit or is already visited from the current active node $\hat{s}$, then this node $\hat{l}$ will be added to the explored set of the active node $\hat{s}$, which is denoted as $\hat{s}^{\otimes}$. Ψ denotes the nodes set on the tree, and $s^{\odot}$ denotes the neighbor nodes set of the nodes s.

After obtaining all the feasible carpooling routes (U), we match the carpooling routes with all the CAVs (V) by minimizing the service time of the ridesharing system. This study assumes that all CAVs are either idle at given deports or en routes. Mathematically, this ridesharing matching problem between AVs and carpool groups is formulated as an optimization model as below.

$$\begin{aligned} \min & \sum_u \sum_v (t_u + t_{uv}) z_{vu} \\ \text{s.t.} & \sum\nolimits_u z_{vu} \leq 1, \quad \forall v \in V \\ & \sum\nolimits_v z_{vu} = 1, \quad \forall u \in U \\ & z_{uv} = \{0, 1\}, \quad \forall u \in U,\ v \in V \end{aligned}$$

where, t_{uv} denotes the service time that CAV v travel to approach a rider in carpool group u. $z_{vu} = 1$ indicates that CAV v is assigned to carpool u. Otherwise, $z_{vu} = 0$. And this model covers the constraints Eq. (2)-(4) in the MIP. Given we assume there are enough CAVs, the greedy algorithm can efficiently solve this optimization model.

Remark 1 (Feasibility): The SCM ensures to generate the feasible solution satisfying the constraints Eq. (2)-(22) in

CSP-MIP. In summary, the construction of the graphic solution space integrates the visiting sequence constraints and time window constraints in Eqs. (9)-(13). The branching procedure and moves ensure that every rider's origin and destination will be visited only once, which are consistent with the constraints in Eq. (5)-(8). The feasibility conditions employed by the cutting procedure make the carpooling routes in a tree satisfy constraints Eqs. (9)-(22). Last, the greedy algorithm will connect the CAV and carpool groups considering the constraints in Eqs (2-4). Therefore, the SCM will obtain an equivalent feasible solution for the CSP-MIP.

Remark 2 (Merits of the CTA): The CTA sustains both feasibility and efficiency. Its exploration procedure may generate multiple carpooling routes serving the same group of riders. The conditions to move the exploration drive the algorithm to find a solution saving most service time for all riders, which is consistent with the objective of the MIP. More importantly, even though the CTA enumerates all feasible solutions, it works very efficiently for three reasons. (i) Each tree searching is limited in the graphic solution space. (ii) The width of a carpooling tree is bounded by the size of carpooling community and the depth of a tree is limited to the capacity of a CAV (e.g., less than 4). Last, the CTA can explore the carpooling route tees from every origin node on G in a parallel manner (see the example in Figure 7) and improve the computation efficiency significantly.

V. Experiments

This study conducts numerical experiments to demonstrate the SCM approach's applicability and merits for solving the large-scale carpooling problem. We first test the sensitivity of some unique setups regarding the carpooling community of the SCM approach. Next, recall that the CSP-MIP model covers a very general carpooling problem setup in literature. Several heuristic algorithms in literature can solve a feasible solution of the CSP-MIP, such as the Naïve ride-matching approach and the Maximum cluster algorithm (MCA) [13]. We want to demonstrate the merits of the SCM approach to three existing ride-matching approaches. The main purpose of the experiments is to examine if the SCM approach can provide a good solution of the CSP-MIP, and if the gain of the computation efficiency will compensate for the loss of the system performance compared to other existing approaches.

A. Experiment Setting up

The numerical experiments are built on the Hardee network, which consists of 44 nodes and 134 links. 100 riders with different O-D pairs are randomly generated in the network. Without loss of generality, we set the length ($\imath$) of the pick-up (or drop-off) time window by randomly selecting a value within the range of [5, 30] minutes for each rider. Then, the pick-up (or drop-off) time window for rider $i \in I$ is set as $[t^{+}(o_i),\ t^{+}(o_i) + \imath\ ]$ (or $[t_{o_i d_i},\ t_{o_i d_i} + \imath]$), where $t^{+}(o_i)$ is the time stamp that the service is required (set as zero in the experiments); $t_{o_i d_i}$ is the shortest travel time from origin o_i to destination d_i according to current traffic. There are five depots in the network for sending out CAVs, which are marked as an orange square in Figure 8. Each experiment was run multiple times and then took the average value to avoid the effect of randomness. For example, in the case of section V.D at rider demand of 20, the standard deviation of system service time under the CTA is 28.5, which is a small number as compared to the average value of system service time 300.2. We did not present the variance for each case but focus on the tendency of the results. The experiments are implemented by MATLAB R2017a. on the HiperGator, the university supercomputer of UF. We run our experiments with requested computation resources 32 CPUs, and RAM: 1 GB per CPU. Below presents the main experimental results and observations.

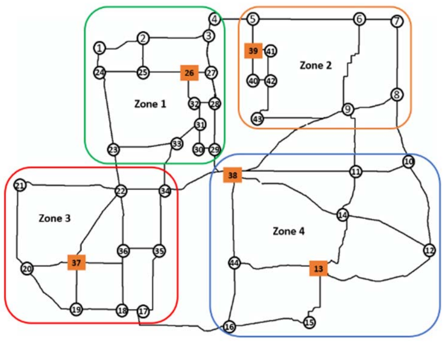


Fig. 8. Hardee network.

B. Validate the Function of SCM

The SCM includes some unique components such as forming carpooling communities and the carpooling tree generation algorithm. They will significantly affect the performance of the SCM. This section wants to test if community detection is necessary and better than the traditional partition method, such as geographical partition. Based on that, we further test the impact of community size on SCM's performance and suggest an appropriate upper bound for the community size. Last, we compare the CTA's computation performance with the existing solver for the MIP, such as Gurobi, to demonstrate the need to develop the CT algorithm even though forming carpooling communities already degrade the problem's scale.

There are different approaches to partition the riders into different carpooling groups. Even though partition will always benefit computation efficiency, but it potentially compromises system performance. A better partition scheme should lead to less system performance loss. These experiments thus demonstrate the merits of the community formation approach in this aspect. Specifically, we compare three cases, in which the SCM respectively integrates three different partition schemes to form the carpooling groups. Case 1: No partition, which considers all riders forming one carpooling community and explores the system optimal carpooling solution. We consider this case as the benchmark to evaluate other partition methods. Case 2: Com-partition, which partitions riders into multiple groups by our community detection and then explores optimal carpooling solutions for each group. Case 3: Geo-partition, which splits riders based on the geographic locations of their

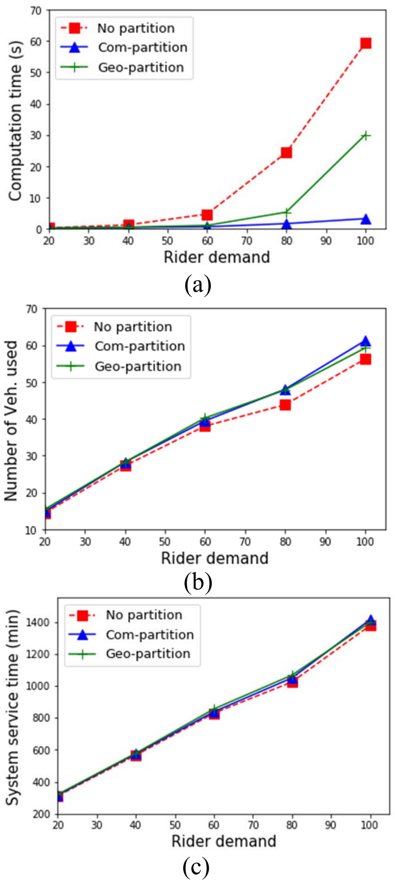


Fig. 9. Performance comparison with and without forming communities before ride-matching among (a) Running time (b) Number of vehicles used (c) System service time.

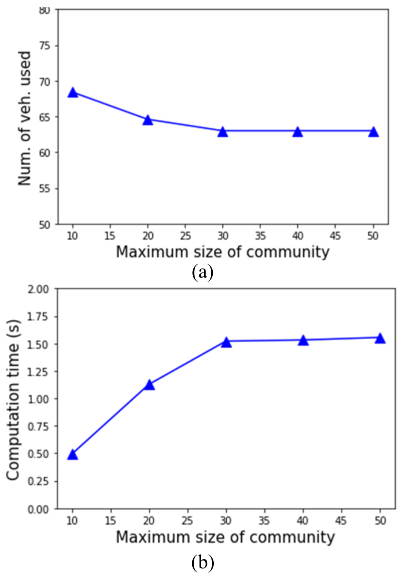


Fig. 10. Test the impact of the carpooling community's size on the system performance and computation time.

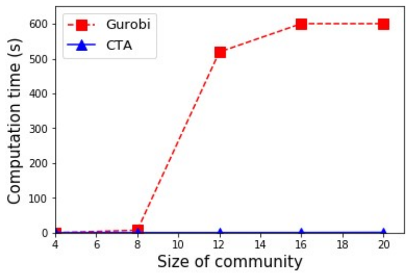
origins, and then explores an optimal carpooling solution for each group. The results in Figure 9 shows that the system performance curve by Com-partition is only slightly worse than by No-partition (8.9% difference in the number of vehicles needed by the service and 2.7% difference in system service time). Thus, we claim that the community formation approach leads to minor system performance loss. Moreover, we can see that the case using Com-partition requests the least computation time among the three cases. For example, when the demand of riders is 100, the case using community detection only took 5.3% of the computation time of the case using no-partition and 10.4% of the case using Geo-partition. In the meantime, we noticed that the case using Com-partition performs slightly worse than the case using Geo-partition in system performance (system service time and the number of vehicles used) even though it presents significant merit in terms of computation efficiency. The key reason is that the experiments using the Geo-partition scheme split riders according to TAZs/districts. It only leads to a small number of groups (e.g., four), and each group evolves many riders. Accordingly, it results in minor system loss, but high computation load. In contrast, our Com-partition presents a good performance in both system operation (system service time and vehicle numbers) and computation efficiency, even though more groups are generated. These results justify the values of using community detection according to the carpooling chances to partition riders before exploring the carpooling solutions for riders in our approach.

C. Sensitivity Test for the Size of Carpooling Community

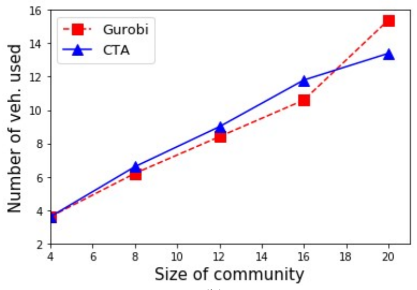
Recall that Section IV.B has discussed that the SCM prefers a group of carpooling communities, each with a small size, since it helps improve the computation efficiency. However, forming a few carpooling communities, each with a relatively large size, will help explore a solution with better system performance. On the other hand, we noticed that each rider could only carpool with a small group of other riders due to the capacity of CAVs and the time window requirements of the riders. This observation implies that increasing the carpooling community's size may not necessarily improve the carpooling solution. Given that both computation efficiency and system performance are important aspects, we first test the community size's sensitivity. To do that, we run the experiments by limiting the community size from 10 to 50 with a step size of 10. The results in Figure 10 show that as the community size increases, the number of carpool vehicles used by the carpool system decrease while the computation time increases. We also noticed the size of the necessary vehicle fleet decreases very mildly after the community size limit passes 20. These results reinforce our opinion that the marginal benefit of improving system performance by increasing community size diminishes after some critical point. However, the computation time continues to increase after that. Therefore, to implement the SCM efficiently, it is proper to limit the community size. The following experiments set the maximum size of a carpooling community as 20.

D. Merits of the Carpooling Tree Generation Algorithm

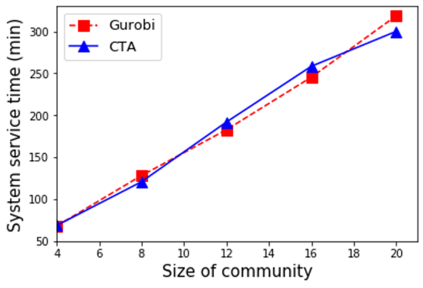
This experiment tests the merit of our CTA by comparing its performance to existing solvers in Gurobi. More exactly,



(a)

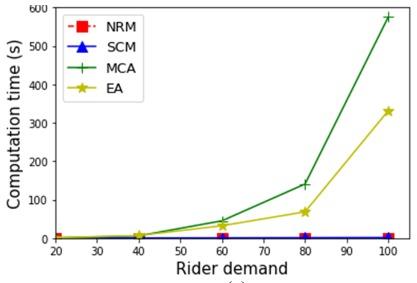


(b)

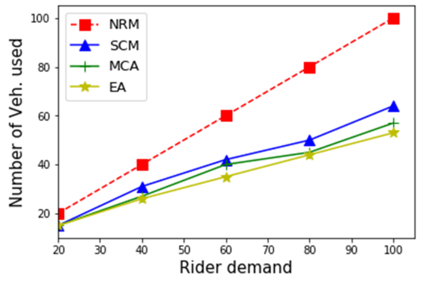


(c)

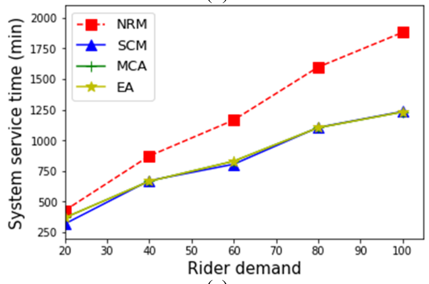
Fig. 11. Performance comparison between the implementation of CTA and Gurobi into riders among (a) Computation time (b) Number of vehicles used (c) System service time.



(a)



(b)



(c)

Fig. 12. Performance comparison between SCM, NRM, MCA, EA among (a) Computation time (b) Number of vehicles used (c) System service time.

we wonder if we can use existing solvers to solve the carpool problem in each community after reducing the size of the CSP by partitioning riders into different carpooling communities. To answer this question, we tested the performances of the CTA algorithm and the Gurobi solver as the size of the community increases from 4 to 20. Given this CSP provides the real-time carpooling route suggestions, the experiments terminate the computation after 10 mins (600 seconds).

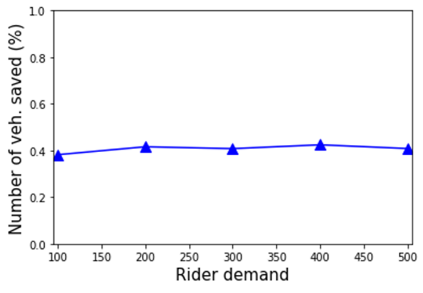
The results in Figure 11 present the solution of the CTA to the best solution that the Gurobi solver can find within 600 seconds. We can see that the CTA greatly improves the computation efficiency while maintaining the same level of system performance as it is compared to the solutions of the Gurobi solver. In particular, as the size of the community reaches 20, the solution of Gurobi suggests a larger vehicle fleet and leads to more system service time than these values in the solution of the CTA. Therefore, we claim that directly solving the MIP using the existing solver cannot produce a practical solution to this CSP problem. This justifies the need for developing the CTA.

E. Comparisons of SCM, NRM, MCA, EA

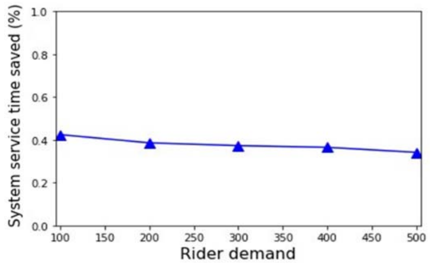
We next compared the SCM with three existing approaches in the literature. We first compare the SCM to the naïve ride-matching approach (NRM) by which the nearest CAV serves every rider. We also compare the SCM to the maximum cluster algorithm (MCA), by which a vehicle tries to find a maximum number of carpooling riders. Thus, MCA solves the single-vehicle-multiple-riders problem with time windows. As multiple vehicles are involved, the system repetitively calls the maximum clustering algorithm [13]. The exact algorithm (EA) is to enumerate all the feasible solutions and find the best match [19].

The results in Figure 12 show that the SCM has a similar computation performance to the NRM but significantly saves system service time and needs fewer vehicles to serve in the system. For example, as the rider demand is 100, the SCM's computation time is 1.29 seconds, which is practical for real-time applications. At the same demand level, we can see that the NRM and SCM respectively lead to the system service time equal to 1880.9 and 1235.7 minutes. The latter is 34.3% less than the former. Besides, the SCM needs 64 vehicles served in the system, saving 36% of vehicles needed by the NRM. Therefore, the SCM outperforms NRM.

Compared to MCA and EA, we can see that the SCM shows minor system performance loss but dramatically improves the computation efficiency. For example, as the rider demand is 100, the MCA needs 57 and the EA needs 53 vehicles to serve all riders with a total service time of



(a)



(b)

Fig. 13. Test the performance of the SCM for the CSP involving a large number of riders.

1242.2 and 1232.5 minutes respectively, while the SCM needs 64 vehicles with a total service time of 1260.7 minutes. However, as we compare their computation time, we found that the MCA took 575.2 seconds and EA took 331.6, while the SCM only needs 1.3 seconds, which is more practical for real-time implementation. Therefore, we conclude that the SCM can locate a near-optimal solution with competitive system performance but significant computation efficiency gain.

F. Scalability

This study further tested the scalability of the SCM by running the experiments with the riders increasing from 100 to 500 by a step size of 100. The system operation performance, including system service time and vehicle saved, is compared to the naïve approach. The results in Figure 13 (a) indicate that the SCM saves about 40.1% of the CAVs needed in the service under different levels of rider demands on average compared to NRM. This merit slightly decreases as the number of riders increases. Figure 13 (b) shows that the SCM reduces by 35.3% of the system service time on average compared to NRM. Thus, It also presents the vantages of SCM to NRM in terms of saving system service time. These merits become stable after the riders are more than 200. Overall, the experiment results confirm the applicability of the SCM to a large-scale CSP.

VI. Conclusion

The carpooling service problem with time window requirements has attracted significant interest from both industry and academia. It has always been challenging to solve. Mathematically, this problem can be modeled as a mixed-integer program (e.g., the CSP-MIP), which is NP-hard. Existing solution approaches often fail to balance the computation efficiency and optimality. This study addresses this challenge by developing the SCM approach, which includes two critical components: 1) strategically partition the riders into multiple smaller rider communities (i.e., decompose the CSP-MIP into multiple sub-CSP-MIPs) to shrink the problem size; 2) explore riders' carpooling routes and optimally match them with CAVs. There are several critical technical issues that need to be addressed for the proposed approach to work efficiently. First, how to partition the riders greatly affects the results of the carpooling solution of riders. The improper partition will significantly compromise the system performance. Moreover, our experiments found that the existing solver cannot efficiently solve a small size of the CSP-MIP to adapt the online application. This study is, therefore, motivated to develop the following methods to address the corresponding difficulties.

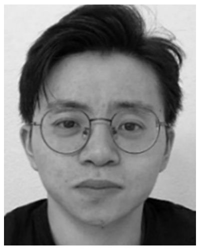
First, to develop efficient decomposition, this study introduces the concept of carpool chance (C2) between riders over a network and uses it as an index to demonstrate the chances of two riders to carpool together. The quantitative method to measure the C2 between riders is developed based on their timing requirements and options of carpooling routes. Based on the identified C2s between riders, a C2 graph is constructed to demonstrate the C2 between each pair of riders over a network. Then, the community detection clustering approach is implemented onto the C2 graph to partition riders into serval smaller communities, which results in high carpool chances between the riders within the same communities but not between riders from different communities. In other words, we strategically decompose the CSP-MIP into multiple sub-MIPs with a minor compromise of system optimality. Secondly, this study transfers the combinatorial problem to a network flow problem by constructing a graphic solution space for each sub-CSP-MIP, which only consists of the feasible connections between the origins and destinations of riders. Built upon that, we develop a carpooling tree generation algorithm (CTA) to explore all feasible carpooling routes among riders and then optimally match them with CAVs to minimize the system service time using the greedy algorithm.

The numerical experiments conducted on Hardee city networks demonstrate the superior performance of the SCM. First of all, our community detection built upon carpooling potential can efficiently decompose the CSP problem for saving 94.7% computation time with minor system optimality compromise. It outperforms Geo-partition. Moreover, the CTA can find a feasible solution for each sub-MIP with a competitive quality but much more efficiently than the existing commercial solver Gurobi. These results confirm the effectiveness of building the graphic solution space and the efficiency of the CTA. Last, this study conducts comparative experiments to demonstrate the vantages of the SCM in terms of the total service time of CAVs and the fleet size to fulfill the service requests. The results show that the SCM saves 37% of the fleet size and 34.3% of CAVs' travel time from the solution of the naïve ride-matching approach. Compared to the maximum cluster algorithm and exact algorithm in literature, it can significantly improve computation efficiency while providing solutions with competitive quality. Overall, the experiments validate the merits and applicability of the SCM.

The presented efforts focus on providing the prescription carpooling solution from a supplier perspective, assuming riders will take the carpooling solution if the time window requirements are satisfied. This assumption can be further extended by considering more comprehensive user preference and behavior, integrating intermodal trips in the carpooling solution, and factoring in the traffic flow uncertainty. In addition, we noticed that existing used data-driven approach to predict the location of riders so that we can well allocate the ride-sharing vehicles to save the waiting time [33], [34]. The proposed CSP approach can also be improved by integrating rider demand prediction. All these considerations involve more realities but significantly complicate the modeling and solution approaches of the CSP. We will investigate them in our future research.

REFERENCES

[1] N. Agatz, "Optimization for dynamic ridesharing: A review," *Eur. J. Oper. Res.*, vol. 223, no. 2, pp. 295–303, 2012.
[2] A. Amey, "A proposed methodology for estimating rideshare viability within an organization, applied to the mit community," in *Proc. TRB Annu. Meeting*, 2011, pp. 1–6.
[3] M. Nourinejad and M. J. Roorda, "Agent based model for dynamic ridesharing," *Transp. Res. C, Emerg. Technol.*, vol. 64, pp. 117–132, Mar. 2016.
[4] M. Li, N. Zheng, X. Wu, and X. Huo, "An efficient matching method for dispatching autonomous vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3013–3018.
[5] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, no. 6, 2004, Art. no. 066111.
[6] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, Feb. 2004, Art. no. 026113.
[7] N. Agatz, A. L. Erera, M. W. P. Savelsbergh, and X. Wang, "Dynamic ride-sharing: A simulation study in metro Atlanta," *Proc.-Social Behav. Sci.*, vol. 17, pp. 532–550, Dec. 2011.
[8] K. Ghoseiri, A. Haghani, and M. Hamed, "Real-time rideshare matching problem," Mid-Atlantic Univ. Transp. Center, Chennai, India, Tech. Rep. UMD-2009-04, 2010.
[9] X. Xing, "SMIZE: A spontaneous ride-sharing system for individual urban transit," in *Proc. German Conf. Multiagent Syst. Technol.* Berlin, Germany: Springer, 2009, pp. 165–176.
[10] K. Bimbraw, "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology," in *Proc. 12th Int. Conf. Informat. Control, Autom. Robot.*, Jul. 2015, pp. 191–198.
[11] S. Winter and S. Nittel, "Ad hoc shared-ride trip planning by mobile geosensor networks," *Int. J. Geograph. Inf. Sci.* vol. 20, no. 8, pp. 899–916, 2006.
[12] H. Tsao and D. Lin, *Spatial and Temporal Factors in Estimating the Potential of 500 Ridesharing for Demand Reduction*. Berkeley, CA, USA: Institute of Transportation Studies, 1999.
[13] L. Häme and H. Hakula, "A maximum cluster algorithm for checking the feasibility of dial-a-ride instances," *Transp. Sci.*, vol. 49, no. 2, pp. 295–310, May 2015.
[14] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, pp. 103–114, Jun. 1996.
[15] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: Indexing micro-clusters for anytime stream mining," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 249–272, 2011.
[16] P. Kalczynski and M. Miklas-Kalczynska, "A decentralized solution to the car pooling problem," *Int. J. Sustain. Transp.*, vol. 13, no. 2, pp. 81–92, Feb. 2019.
[17] R. Baldacci, V. Maniezzo, and A. Mingozzi, "An exact method for the car pooling problem based on Lagrangean column generation," *Oper. Res.*, vol. 52, no. 3, pp. 422–439, Jun. 2004.
[18] W. M. Herbawi and M. Weber, "A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time Windows," in *Proc. 14th Int. Conf. Genetic Evol. Comput. Conf.*, 2012, pp. 385–392.
[19] M.-K. Jiau and S.-C. Huang, "Services-oriented computing using the compact genetic algorithm for solving the carpool services problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2711–2722, Oct. 2015.
[20] P. Minett and J. Pearce, "Estimating the energy consumption impact of casual carpooling," *Energies*, vol. 4, no. 1, pp. 126–139, Jan. 2011.
[21] S. Seyedabrishami, A. Mamdoohi, A. Barzegar, and S. Hasanpour, "Impact of carpooling on fuel saving in urban transportation: Case study of Tehran," *Proc.-Social Behav. Sci.*, vol. 54, pp. 323–331, Oct. 2012.
[22] J. Lo and S. Morseman, "The perfect uberPOOL: A case study on trade-offs," in *Proc. Ethnograph. Praxis Ind. Conf.*, Jan. 2018, pp. 195–223.
[23] Y. Li and S. H. Chung, "Ride-sharing under travel time uncertainty: Robust optimization and clustering approaches," *Comput. Ind. Eng.*, vol. 149, Nov. 2020, Art. no. 106601.
[24] S. Chen, H. Wang, and Q. Meng, "Solving the first-mile ridesharing problem using autonomous vehicles," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 35, no. 1, pp. 45–60, Jan. 2020.
[25] A. Tafreshian and N. Masoud, "Trip-based graph partitioning in dynamic ridesharing," *Transp. Res. C, Emerg. Technol.*, vol. 114, pp. 532–553, May 2020.
[26] H. Xu, J. Pang, F. Ordánez, and M. Dessouky, "Complementarity models for traffic equilibrium with ridesharing," *Transp. Res. B, Methodol.*, 81, pp. 161–182, Apr. 2015.
[27] S. Feigon and C. Murphy, "Shared mobility and the transformation of public transit," TRB Conf. Project J-11, Task 2, 2016.
[28] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time Windows," *Eur. J. Oper. Res.*, vol. 54, no. 1, pp. 7–22, 1991.
[29] W. P. Nanry and J. W. Barnes, "Solving the pickup and delivery problem with time Windows using reactive tabu search," *Transp. Res. B, Methodol.*, vol. 34, no. 2, pp. 107–121, 2000.
[30] H.-F. Wang and Y.-Y. Chen, "A genetic algorithm for the simultaneous delivery and pickup problems with time window," *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 84–95, Feb. 2012.
[31] R. Baldacci, E. Bartolini, and A. Mingozzi, "An exact algoritFhm for the pickup and delivery problem with time Windows," *Oper. Res.* vol. 59, no. 2, pp. 414–426, 2011.
[32] L. Grandinetti, F. Guerriero, F. Pezzella, and O. Pisacane, "The multi-objective multi-vehicle pickup and delivery problem with time Windows," *Proc.-Social Behav. Sci.*, vol. 111, pp. 203–212, Feb. 2014.
[33] G. Guo and T. Xu, "Vehicle rebalancing with charging scheduling in one-way car-sharing systems," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 20, 2020, doi: 10.1109/TITS.2020.3043594.
[34] G. Guo and Y. Xu, "A deep reinforcement learning approach to ride-sharing vehicles dispatching in autonomous mobility-on-demand systems," *IEEE Intell. Transp. Syst. Mag.*, early access, Apr. 1, 2020, doi: 10.1109/MITS.2019.2962159.



Wang Peng received the B.S. degree in civil engineering from the Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2015, and the Ph.D. degree from the University of Florida in 2021. His research interests include networking modeling, coordinated traffic management, carpooling, and multimodal transportation systems.

Lili Du (Member, IEEE) received the B.S. degree in mechanical engineering from Xi'an Jiaotong University, China, in 1998, the M.S. degree in industrial engineering from Tsinghua University, China, in 2003, and the Ph.D. degree in decision sciences and engineering systems with a minor in operations research and statistics from Rensselaer Polytechnic Institute (RPI), USA, in 2008. She is currently an Associate Professor with the Department of Civil and Coastal Engineering, University of Florida (UF). Before joining at UF, she worked as an Assistant and then an Associate Professor with the Illinois Institute of Technology from 2012 to 2017 and a Post-Doctoral Research Associate with the NEXTRANS UTC Center, Purdue University, from 2008 to 2012. Her research is characterized by applying operations research, network modeling, machine learning, and big data analytics methods into transportation system analysis and network modeling. Her current research focuses on connected and autonomous vehicle routing and platooning control, electric vehicles, resilient infrastructure networks, traffic flow analysis and prediction, mobility as a service, and sustainable multimodal transportation systems.