Ride-matching and Routing Optimisation: Models and a Large Neighbourhood Search Heuristic

June 7, 2018

Abstract

This paper considers a ridesharing problem on how to match riders to drivers and how to choose the best routes for vehicles. Unlike the others in the literature, we are concerned with the maximization of the average loading ratio of the entire system. Moreover, we develop a flow-dependent version of the model to characterize the impact of pick-up and drop-off congestion. In another extended model we take into account the riders' individual evaluation on different transportation modes. Due to the large size of the resulting models, we develop a large neighbourhood search algorithm and demonstrate its efficiency.

Keywords ride-matching and routing, ridesharing systems, large neighbourhood search

1 Introduction

Ridesharing refers to a transportation mode in which individual travellers share a vehicle for similar itineraries and time schedules. In essence, ridesharing entails the participation of one or more riders (peer customers) to share a vehicle (typically a car) together with the driver (peer provider) when travelling from start points to destinations. The benefits of the ridesharing mode include the split of travelling costs such as gas, toll, and parking fees among individual travellers, and the reduction of congestion and pollution to the public. Despite these benefits, the ridesharing mode is still not a regular transportation alternative, due to the lack of efficient methods to coordinate itineraries and schedules. In the recent decades, technological advances including the global positioning systems, the mobile internet, and social networking create the "critical mass" for the potential prevalence of ridesharing. With the advance of these technologies, a number of matching agencies emerged to provide diverse ridesharing services to travellers. This is mainly stimulated from the development of various ridesharing platforms that create a "pool" for connecting peer riders' travelling demands and peer drivers' services. For instance in China, which has some of the most congested cities in the world, a leading internet firm Tencent uses its ridesharing app, Didi Dache (Honk Honk Taxi), as a strategic tool for the ridesharing market penetration.

Along with this trend multiple decision making problems have emerged, most of which are concerned with the development and optimisation of driver-rider matching. One of the most extensive surveys on the state of the ridesharing systems and the future directions was provided by Furuhata et al. [22], who pointed out that the rider matching is one of the main challenges. In [22], the authors investigated and classified a set of representative matching agencies, and identified two main taxonomic criteria for ridesharing systems: *primary search criteria* and *target market*. With the aid of the information technologies, *trip planning* becomes one of the common and main functions of matching agencies to support matching between drivers and riders. Each of the drivers and riders lists their offers and requests for ridesharing. while the active matching tries to find potential partners. One of the key issues in a ridesharing platform is how to optimise the success rate of ride-matching with a reasonable notice period. The core in the solution of this issue is to devise a matching algorithm that can recommend a driver with the most appropriate riders in align with his or her itinerary and time schedule. Its aim is to enhance the ridesharing efficiency and keep the willingness of drivers to participating in ridesharing, so as to maximize the seat utilization in the vehicles with tolerable impacts on drivers' travelling routes and time schedules.Instead of peer-to-peer communication, ridesharing platforms have their scale advantages and computational advances in providing tremendous matching options between the drivers and riders. A "smarter" solution from the whole ridesharing system's perspective is usually superior to the matching activities conducted by a driver and several riders themselves in both economic and time efficiencies.

The most recent review on the ride-matching optimisation problems was undertaken by Agatz et al. [2]. These problems consider how to determine the routes and schedules of the vehicles (including how to assign riders to drivers) in the presence of conflicting objectives, such as maximizing the number of serviced riders, minimizing the operating cost or minimizing the rider inconvenience. Given these objectives, most of the ridesharing systems prefer to give drivers sufficient time flexibility so that they may be willing to provide rides to several riders along their itineraries either one after another or simultaneously for a portion of time. By doing so, the system makes the effort to achieve ridesharing as many as possible. The value of a ridesharing plan is measured by the combination of the gain to riders due to cost savings and the loss to drivers due to additional travelling time.The above facts show that the decision process in ridesharing systems is similar to dial-a-ride problems/models (DARP). One of the important differences is that in a dial-a-ride system all vehicles typically operate out of one or more depot locations, whereas in a ridesharing system each driver may have a unique origin-destination (OD) pair (see [2]). Since the riders are usually independent or partially independent in these systems, they are not obligated to accept ridesharing arrangements that they do not like. Therefore, drivers' route preferences or at least the locations of their origins and destinations need to be accounted for when matching drivers and riders in a ridesharing system. This motivates the research of adapting the existing DARP models for solving the ride-matching optimisation problems. To the best of our knowledge, the first variant of dial-a-ride models for the ride-matching optimisation problems was exploited by Baldacci et al. [5], who considered a matching problem in a car pooling service organized by a large company to encourage its employees to pick up colleagues while driving to/from work to minimize the number of private cars travelling to/from the company site. In this first attempt, some assumptions in the dial-a-ride problems are reserved, such as identical vehicles, the same terminal for all drivers, and the exclusion of the seat utilization in the objective. The proposed model and exact algorithms in [5] are adequate to deal with some of the complexities in this real world problem.

In this paper we consider a ride-matching problem and the associated routing of vehicles. We make our contributions from three perspectives. Firstly, unlike the others in the literature, we consider a different objective, which is to maximise the average loading ratio of the vehicles in the whole system. This target reflects the social expectation on the use of ridesharing systems, as a higher ratio means better utilization of the vehicles. Intuitively a higher ratio implies that more riders should be allocated to each driver, which however might result in a situation where the drivers need to travel longer distances to pick-up and drop-off the matching riders, leading instead to a low loading ratio for their itinerary. We propose a non-linear programming model to formulate this problem, which is then transformed to an equivalent integer program.

Secondly, we extend the model to capture the flow-dependent features in real ridesharing systems. Specifically, we take into account the impact of pick-up and drop-off congestions and delays to the system performance. We also make an attempt to include the riders' individual evaluation into the ride-matching process. Indeed, in practice how the drivers and the riders match each other highly depends on their evaluations of ride-matching recommendations. As far as we know, both are the first attempts in the literature and contribute to the mathematical framework of ridesharing by incorporating real behaviours of riders into the optimisation of the system performance.

Finally, to address the complexity introduced to the proposed models, we establish a variant *large neighbourhood search (LNS)* framework to improve the solution efficiency. A new demand removing method and a randomization method for demand reassignment are proposed and integrated into this framework.

The rest of the paper is structured as follow. In section 2, we review the related works to the models and the heuristic solution methods to ridesharing problems in the literature. In section 3, we propose a variant DARP model to the ride-matching and routing problem concerned and investigate a linear reformulation of this model. We extend this model to capture flow-dependent features and include riders' individual evaluation into the ride-matching function in section 4. In section 5, we develop a solution algorithm within the LNS framework, which is tested in section 6 to a number of randomly generated problem instances and its performance is compared against the standard solvers. Finally, a conclusive discussion is presented in section 7.

2 Literature Review

In this section, we review the related literature to our work, with the studies on ridesharing, dial-a-ride problems and heuristics being the three main areas.

2.1 Ridesharing

As a rather new transportation mode, one of most important issues that has been well studied in ridesharing is to identify the key properties in ridesharing systems. A comprehensive review on the conditions for a successful ridesharing system can be found in Agatz et al. [2]. In the literature, technological advances in both hardware and software have been recognized as the key enablers for an efficient ridesharing system. In particular, the ubiquity of Internet-enabled mobile devices plays a critical role in practical dynamic ridesharing [24] and [9]. Moreover, a sustainable ride sharing system, as argued by Raney [50], must be able to attract riders and drivers, and to reach a critical mass of participants. A few approaches were discussed in an article by Gaynor [19]. Apart from improving the technology, the system provider could aim at increasing marketing efforts and engaging people with societal/behavioural approaches.

Though the prevalence of ridesharing systems is recently emerged and accepted by the public, the first regulation policy to organize ridesharing was in place by the U.S. government during WWII for fuel conservation, and the second surge of ridesharing methods dates back to the 1970s as a result of the oil crisis. However, the traditional ridesharing mechanism is not sufficient to accommodate the unconventional schedules of today's ridesharing demand, where many commuters will only respond to flexible commuting options [30] and the optimisation approaches are not necessarily involved. With the growth of the number of drivers and riders involved in ridesharing systems, how to efficiently match the ridesharing supplies and demands becomes a bottleneck in the acceptance of these systems. The theoretical researches are urgently demanded to cope with the complexity in the realization of the core ride-matching functions both in modelling and computation. A substantial volume of researches have been conducted to propose schemes to improve the efficiencies, or more precisely, the likelihood of ride-matching. For example, to increase the ride-matching rate, Masoud et al. [36] recently proposed a peer-to-peer ride exchange mechanism. With such a mechanism the riders who could not be matched can buy a previously matched rider's trip, if an alternative itinerary is available for the seller.

Optimisation algorithms have been playing an important role in the ride-matching problems. The simplest variant of such problems only allocates to each driver a single rider, which can be formulated as a maximum-weight bipartite matching problem (Agatz et al. [2]). In practice each driver might be willing to serve more than one riders, and the resulting problems are much more complicated. Baldacci et al. [5] proposed two integer programming formulations to the car pooling problem, which are then solved by both an exact and a heuristic methods. Another important variant is the dynamic ride sharing problems where the drivers and the riders enter and leave the system continuously over time. Agatz et al. [1] considered such a problem in a simulation study of Metro Atlanta. The uncertainty is dealt with by a rolling horizon approach. For other similar works see Winter and Nittel [64] and Xing et al. [65]. Note that we restrict our attention to the static ride sharing problems in this work.

Agatz et al. [2] pointed out two limitations of the current works on the optimisation approaches. Firstly, the existing approaches may not be capable of solving instances with realistic sizes and thus a clear need for faster and practically implementable approaches. Secondly, along with the increased urban traffic network, the scale of a ridesharing problem could be extremely large, which prevents the applicability of most of the existing ride-matching approaches that address the problem from a centralized perspective. In this paper, we also explore a decentralized approximate framework by splitting the entire traffic network into several subsets in the numerical test section 6.

2.2 Dial-a-Ride Problems

From the methodological perspective, the relationship between ridesharing systems and dial-a-ride problems has been recognized by many researchers (see [41] and [5]). As mentioned in section 1, several variants of DARP models have been adopted to solve the ridesharing problems.

The dial-a-ride problems are also close to the well-known pick-up and delivery problems with time windows (PDPTW). As a generalization of vehicle routing problems (VRP), the classical PDPTW is characterized by paired pick-up and delivery locations, i.e., each pick-up location is associated with a particular delivery location and vice versa [39]. The study on PDPTW and dial-a-ride problems date back, to our best knowledge, to Psaraftis [48] for a problem with a single vehicle. A dynamic programming method was employed to solve PDPTW with a small number of delivery demands (less than ten) in the traffic network. Later on, Psaraftis [49] extended the algorithm to solve the problem with hard time windows. One of the early applications of DARP is the door-to-door transportation services for sick or disabled people (see [34] and [60]).

After that, the PDPTW and DARP have been further generalized and received more attention in Solomon and Desrosiers [59] and Savelsbergh and Sol [56]. The demand for the exact solution methods grew along with these researches. The dynamic programming by Desrosiers et al. [16] is the first approach used for solving larger problem instances. Shortly after that, Dumas et al. [18] successfully solved PDPTW with the number of delivery demands up to 55 by a branch-and-pricing approach. A branch-and-cut algorithm was developed by Lu and Dessouky [32] to optimally solve the integer-programming formulation of the problem. Cordeau [12] and Ropke et al. [51] explored an alternative formulation and a branch-and-cut algorithm to optimally solve the problem. A new branch-and-cut-and-price algorithm was later on proposed by Ropke and Cordeau [52], where, according to the computational experiments, outperformed the branch-and-cut algorithm of Ropke et al. [51]. In the most recent development of the exact algorithm, Cherkesly et al. [10] explored a PDPTW with last-in-first-out loading constraints and designed a branch-pricing-cut algorithm to solve it.

The dynamics in PDPTW nowadays has gained more and more attention, in which transportation requests are generated over time in contrast to the static cases where the entire problem is known beforehand (see Savelsbergh and Sol [57]). The modeling framework and the algorithms for dynamic vehicle routing were studied by Fabri and Recht [21] and Gendreau et al. [20], where a dynamic pick-up and delivery vehicle routing with several time windows and waiting times was proposed in [21] and a neighbourhood search heuristic was exploited in solving a dynamic vehicle dispatching problem in [20]. Along with the increase of complexity in dynamical systems, methods addressing these dynamics are further studied in more recent works, such as, Mitrović-Minić and Laporte [37] in evaluating waiting strategies to deal with these dynamics, Sáez et al. [54] in designing hybrid adaptive predictive control strategies for a dynamic problem.

Two features of ridesharing make our study significantly different from the existing literature. Firstly, for ridesharing in urban cities, the flow-dependent model is critical. For instance, in the peak time period drivers might be much sensitive to the cost of congestion caused by too many ridesharing activities of pick-up and drop-off at some locations. Secondly, how drivers and riders match each other through the system is highly dependent upon their evaluation of the routes, schedules, etc. Therefore, the choice behaviour pattern needs to be captured to measure the impact from the individual evaluation onto the performance of the ride-matching plans recommended by the ridesharing system. More importantly, the inter-relationship between ridesharing, modal-split (auto, public transit, ridesharing etc.) and congestion is very important, and should be treated in the ridesharing problems. In our study, we make effort to include these two features into consideration.

2.3 Heuristics

The ridesharing and the closely related DARP or PDPTW problems that can be solved by exact methods are proven to be limited. Particularly, the trip planning problem in the ridesharing is NP-hard in the strong sense as it includes a special case of the VRP with unit customer demand (see [4] and [29]).

Therefore a large body of literature turns to heuristic methods, which are more efficient in solving large scale problems. The idea of using heuristic methods to solve PDPTW is not new. An early study on the heuristics for PDPTW was performed by Jaw et al. [25], where they required that users can only specify either the pick-up time or the delivery time. Along this direction, Potvin and Rousseau [46] improved the insertion operations in the heuristic proposed in [25] and added two new phases. Tabu search framework is another important heuristic that has been well-implemented for solving PDPTW and dial-a-ride problems. The first work using this approach is due to Nanry and Barnes [38] for a subclass of PDPTWs, in which problems with up to 50 demands have been designed and used as benchmarks for the test of the efficiency of the algorithms. Li and Lim [31] implemented a simulated annealing procedure to this tabu search framework to increase the size of solvable PDPTW up to 100 demands. Lau and Liang [28] also used a tabu search framework to solve PDPTW and investigate several different approaches in routing. The most recent research on tabu search for dial-a-ride problems was from Cordeau and Laporte [13]. They described a tabu search heuristic and proposed a procedure for neighbourhood evaluation that adjusts the number of visit times of the vertices on the routes so as to minimize route duration and ride times. This is also the first work connecting tabu search heuristics and the concept of neighbourhood search, where the latter is proved to be very efficient in the solution procedure. Since then, the research on how to create effective neighbourhood search heuristics started to obtain wider attention. Bent and van Hentenryck [7] studied a two-stage mixed algorithm with a simulated annealing method at the first stage to reduce the number of vehicles for delivery and a large neighbourhood search to minimize the total distance for cargo delivery. Based on the above methods, Ropke and Pisinger [53] proposed an adaptive large neighbourhood search (ALNS) algorithm and investigated several demand remove and route reconstruction operations to improve the efficiency. Particularly for static multi-vehicle dial-a-ride problems, a competitive variable neighbourhood search (VNS)-based heuristic was analysed by Parragh et al. [40], which allows intermediate deteriorating moves. The other heuristics applied to PDPTW and dial-a-ride problems include the population algorithm by Cherkesly et al. [11], the parallel regret insertion heuristic by Diana and Dessouky [17], and the ant colony optimisation by Catay [8].

The heuristic methods for DARP and PDPTW have been systematically reviewed in the literature (Laporte [29], Vidal et al [55] and Pillac et al [39]). According to the surveys, these heuristic methods, which focus on theoretical frameworks on DARP and PDPTW models, cannot be directly applied to ridesharing problems. As a consequence, there exist an increasing number of studies on heuristic approaches of solving dynamic multi-vehicle dial-a-ride problems (see Madsen et al.[35]), though the number of studies are still small and the size of the problems solved are up to several hundreds. It is worth mentioning that the extension in [35] covered the dial-a-ride problems with time windows, multiple capacities, and diversified objectives.

3 Model

This section proposes a variant DARP model for trip planning in a ridesharing problem. Unlike most DARP models, to provide a ridesharing solution, two issues have to be solved simultaneously: ride-matching to automatically assign appropriate riders to drivers and vehicle routing for drivers to pick-up and deliver the riders and complete his or her own itinerary. In our model, an optimal solution is attained by jointly solving these two issues.

A ridesharing demand can be described by a specific pair of pick-up and delivery locations and measured by the number of riders to be picked up. We have assumed that all the riders at a pick-up location have the same drop-off location and thus are belong to a single demand. This assumption can be readily relaxed by splitting the riders from the same pick-up locations with different destinations into multiple demands, and adding “virtual” links between the locations of these demands. In the model, every driver's itinerary can be viewed as a ridesharing demand as well where the pick-up and delivery locations are the start and end points of their itineraries.

Regarding the time-constrained feature of riders and drivers, we define a hard time window for each ridesharing demand, that is, the time when a rider or a driver must be picked-up or start and when they are expected to be delivered to or arrive at the destination. This assumption can be easily relaxed by setting the corresponding time windows as $[0, +\infty)$. Moreover, in our model the vehicles are allowed to arrive at a location before the start of the time window, and wait until the time window opens to commence the pick-up service. We further assume that the service time at each location is zero, each pair of locations have symmetrical distance and driving time, and the vehicle capacity is limited.

The model is designed to recommend feasible ridesharing demands en route for a specific driver, as long as the capacity constraint and time window constraint are satisfied. Note that due to the limited number and geographical conditions of the drivers, it is possible that the model cannot attain suitable drivers for every rider. Therefore, satisfying all ridesharing demands is not compulsory in our model.

3.1 The Formulation

Consider a ridesharing system with the task of matching each of the drivers to a set of riders. We denote by $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ the traffic network based on which the ridesharing system is operated. Here $\mathcal{N}$ is the node set of all possible locations that the ridesharing demand might occur, i.e., the locations of potential drivers or riders. We index the nodes (locations) in $\mathcal{N}$ by $i = 1, 2, \cdots, 2m + 2n$, where m and n are the number of nodes where the drivers and riders might be located, respectively. It comprises four subsets. Denote by $\mathcal{O} := \{1, 2, \cdots, m\} \subset \mathcal{N}$ the subset of nodes where the drivers start their itineraries. After that, a driver will pick up the assigned riders by visiting the corresponding nodes in subset $\mathcal{O}' := \{m + 1, \cdots, m + n\}$. If a driver serves a demand at node $m + i$, they are forced to visit the node where the destination of this demand is along their itinerary. Denote by $\mathcal{D}' = \{m + n + 1, \cdots, m + n + n\}$ a subset of the destination nodes of riders, and therefore node $m + n + i$ is the destination location of the riders who are picked up at node $m + i \in \mathcal{O}'$. After finishing the services for all the assigned demands, the driver from node $i \in \mathcal{O}$ will finally arrive at their own destination node that is denoted by $m + 2n + i$. We use $\mathcal{D} = \{m + 2n + 1, \cdots, m + 2n + m\}$ to denote the set of these nodes. Finally, we denote by node $\hat{n}$ the end depot of all the vehicles. Each vehicle is virtually returned to depot $\hat{n}$ after leaving the corresponding destination node of their drivers. We define the set of all precedent nodes as $\mathcal{N}^+ = \mathcal{N} \cup \{0\}$ and the set of all successive nodes as $\mathcal{N}^- = \mathcal{N} \cup \{\hat{n}\}$. For the connections in the ridesharing system, we denote the set of all arcs by $\mathcal{A}$ that is a subset of $\mathcal{N}^+ \times \mathcal{N}^-$. An arc $(i, j) \in \mathcal{A}$ is defined as a link connecting node $i \in \mathcal{N}^+$ and node $j \in \mathcal{N}^-$.

We call by *ride-matching* function the decision process to recommend the riders to the right drivers. The core functionality of the real ride-matching system is to recommend a number of riders to each driver so as to achieve the predefined targets. The set of decision variables $\{x_{ij}^k, k = 1, 2, \cdots, m; (i, j) \in \mathcal{A}\}$ are used to represent the ridesharing system's recommendations on the sequence of riders to be picked up by a specific vehicle, where $x_{ij}^k = 1$ implies that after leaving node i, vehicle k should go to node j to pick up or deliver the corresponding riders, and $x_{ij}^k = 0$ otherwise. Note that we have $x_{0j} = 1$ for all $j \in \mathcal{O}$, which means that every vehicle is enforced to visit the node of the corresponding driver after virtually leaving node 0. A route can be constructed from the values of x_{ij}^k. For instance, if we have in the decision $x_{0k}^k = 1, x_{k(m+i)}^k = 1, \cdots, x_{j(m+n+i)}^k = 1, \cdots, x_{(m+n+i')(m+2n+k)}^k = 1, x_{(m+2n+k)\hat{n}}^k = 1$, the ridesharing system recommends the driver of vehicle k to visit the nodes $m+i, \cdots, j, m+n+i, \cdots, m+n+i', m+2n+k$ in sequence.

The other key parameters including travelling times and distances between any two connected nodes, the capacity of each vehicle, and etc., are listed in Table 1.

In addition, we use the time window to represent the riders' and the drivers' tolerance in time. Note that we define the time window at depot 0 being $[0, \infty)$ and the distances between depot 0 and the starting node of every driver's itinerary being 0. Following the itineraries, we define the time window for the driver at node k as $[a_k, b_k]$ with $b_k > a_k > 0$ to represent the driver's tolerance on times for starting their itinerary. It is not difficult to see that we can let $b_k = a_k$ if driver k is inflexible with respect to the start time.

To model the riders' tolerance in their start times, we define the earliest and the latest times for each rider to be picked up from node $i \in \mathcal{O}$ by $[a_i, b_i]$. Again we can split the riders at the same node with different starting time tolerances into multiple virtual nodes. The practical interpretation is as follows. The earliest time the riders at node i show up is a_i and these riders cannot wait later than b_i. Symmetrically, we extend the definition of time windows to the nodes in $\mathcal{D}$ and $\mathcal{D}'$ as the riders' and drivers' tolerances on the arrival times of their destinations. Denote by s_i^k the vehicle k's arrival time at node i, and $\mathbf{s} := \{s_i^k, k \in \mathcal{O}, i \in \mathcal{O}'\}$. Without loss of generality, we set $s_k^k := 0$ for $k \in \mathcal{O}$. Any feasible solution needs to have $s_i^k \in [a_i, b_i]$.

Besides the arrival times, we need another set of variables on the number of people in every vehicle for the capacity constraints. We denote the capacity of vehicle k by c_k, and an integer variable Q_i^k to denote the number of people (including the driver and all the riders) in vehicle k when it leaves node i. If $i \in \mathcal{O}'$, then we have $Q_i^k - Q_j^k = q_i$ where j is the proceeding node visited by vehicle k, and q_i is the number of riders to be picked up at node i that is known prior to ride-matching. If $i \in \mathcal{D}'$, then we have $Q_i^k - Q_j^k = -q_i$ and q_i is the number of the riders to be delivered by vehicle k at node i. In the model, we need to guarantee that $Q_i^k \leq c_k$ at all node $i \in \mathcal{N}$ with $x_{ji}^k = 1$. It is worth extending these constraints to the nodes where the drivers start their itineraries. That is, the model requires $Q_i^k - Q_j^k = q_i := 1$ for $i \in \mathcal{O}$ and $Q_i^k - Q_j^k = q_i := -1$ for $i \in \mathcal{D}$, where a driver starts and terminates their itineraries. In addition, we define $q_0 = q_{\hat{n}} = 0$ for the virtual depots.

We summarize all notation used in the model in Table 1.

Table 1: Notation

	notation	meaning
parameters	m	the number of nodes where each driver locates, i.e., the number of the vehicles;
	n	the number of nodes where ridesharing demands might locate;
	i, k	we use i to index the node in set $\mathcal{O}$ and $m+2n+i$ the corresponding node in set $\mathcal{D}$; and we use k and $n+k$ to index the node in set $\mathcal{O}'$ and $\mathcal{D}'$;
	q_i	$q_i > 0$ indicates the number of riders to be picked up at node i, and $q_i < 0$ indicates the number of riders to be dropped off the vehicle at node i;
	a_i, b_i	the start and end times of the time window at node i;
	d_{ij}	the distance of the arc $(i,j) \in \mathcal{A}$;
	t_{ij}	the travelling time along the arc $(i,j) \in \mathcal{A}$;
	c_k	the capacity (the number of seats) of vehicle k;
	σ	the target loading ratio of the system;
	M	a large positive number;
	$\mathcal{N}$	the set of nodes, i.e., $\mathcal{N} := \{1, 2, \cdots, 2m+2n\}$;
	$\mathcal{V}$	the set of vehicles, i.e., $\mathcal{V} := \{1, 2, \cdots, m\}$;
variables	x_{ij}^k	binary variables, $x_{ij}^k = 1$ if arc (i,j) is included in the route of vehicle k, $x_{ij}^k = 0$, otherwise;
	s_i^k	the time when vehicle k arrives at node i;
	Q_i^k	the total number of people on-board when vehicle k leaves node i.

The optimisation problem faced by the ridesharing system operator can be formulated as follows.

$$
\text{(M1)} \quad \max \quad \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{V}} x_{ij}^k d_{ij} Q_i^k - \sigma \sum_{k \in \mathcal{V}} c_k \left(\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} x_{ij}^k d_{ij} \right) \tag{3.1}
$$

$$
\text{s.t.} \quad x_{0k}^k = 1, \qquad \forall k \in \mathcal{O}, \tag{3.2}
$$

$$
\sum_{j \in \mathcal{O}' \cup \mathcal{D}} x_{kj}^k = 1, \qquad \forall k \in \mathcal{V}, \tag{3.3}
$$

$$
\sum_{k \in \mathcal{V}} \sum_{j \in \mathcal{N}^+} x_{ji}^k \leq 1, \qquad \forall i \in \mathcal{O}', \tag{3.4}
$$

$$
\sum_{i \in N} x_{il}^k - \sum_{j \in N} x_{lj}^k = 0, \qquad \forall l \in \mathcal{O}' \cup \mathcal{D}', \forall k \in \mathcal{V}, \tag{3.5}
$$

$$
\sum_{j \in \mathcal{N}^-} x_{lj}^k - \sum_{i \in \mathcal{N}^+} x_{i,n+l}^k = 0, \qquad \forall l \in \mathcal{O}', \forall k \in \mathcal{V}, \tag{3.6}
$$

$$
s_{n+i}^k - s_i^k \geq 0, \qquad \forall i \in \mathcal{O}', \forall k \in \mathcal{V}, \tag{3.7}
$$

$$
\sum_{j \in \mathcal{O} \cup \mathcal{D}'} x_{j(m+2n+k)}^k = 1, \qquad \forall k \in \mathcal{V}, \tag{3.8}
$$

$$
x_{(m+2n+k)\hat{n}}^k = 1, \qquad \forall k \in \mathcal{O}, \tag{3.9}
$$

$$
s_i^k + t_{ij} - M\left(1 - x_{ij}^k\right) \leq s_j^k, \qquad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}, \tag{3.10}
$$

$$
a_i \leq s_i^k \leq b_i, \qquad \forall i \in \mathcal{N}, \forall k \in \mathcal{V}, \tag{3.11}
$$

$$
Q_j^k \leq c_k \sum_{i \in \mathcal{N}^-} x_{ij}^k, \qquad \forall j \in \mathcal{N}, \forall k \in \mathcal{V}, \tag{3.12}
$$

$$
Q_i^k + q_j - M\left(1 - x_{ij}^k\right) \leq Q_j^k, \qquad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}, \tag{3.13}
$$

$$
Q_i^k + q_j + M\left(1 - x_{ij}^k\right) \geq Q_j^k, \qquad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}, \tag{3.14}
$$

$$
Q_k^0 = 0, \qquad \forall k \in \mathcal{V}, \tag{3.15}
$$

$$
Q_k^{\hat{n}} = 0, \qquad \forall k \in \mathcal{V}, \tag{3.16}
$$

$$
x_{ij}^k \in \{0, 1\}, \; s_i^k \geq 0, \; Q_i^k \geq 0, \qquad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}. \tag{3.17}
$$

We first explain in detail each of the constraints. Constraints (3.2) require that virtually every vehicle must visit the node where the corresponding driver locates immediately after leaving the depot $i = 0$. Constraints (3.3) enforce that when a driver leaves the start node of his or her itinerary, they could be recommended to pick-up one or more riders from a node in $\mathcal{O}'$ or directly go to their destination node in $\mathcal{D}$. Constraints (3.4) show that the ridesharing system assigns at most one driver to serve a demand, where the inequalities imply that the ride-matching solutions are not compulsory to meet all ridesharing demands. Constraints (3.5) are called *balance* constraints that ensure every vehicle leaves a node after picking up or delivering riders if this node is not the driver's start or destination location. Constraints (3.6) require that a vehicle which serves the demand at node $l \in \mathcal{O}'$ must drop-off these riders at node $n+l \in \mathcal{D}'$, and constraints (3.7) ensure that the arrival time at node $l \in \mathcal{O}'$ must be earlier than the arrival time at the destination node $n + l$. Constraints (3.8) require that the proceeding node before a driver's destination in the recommended route must be either a node to drop-off riders or a node where their itinerary starts from. Constraints (3.9) enforce that every vehicle must return virtually to the depot $\hat{n}$ immediately after it leaves the driver's destination. By constraints (3.10) the precedence order for each itinerary is maintained. Constraints (3.11) ensure that the arrival time of a vehicle at a specific node must be within the time window of the passengers waiting at this node.

Constraints (3.12) to (3.16) are introduced to enforce the vehicle capacity and the flow balance of all passengers. Constraints (3.12) mean that if node j is visited by vehicle k (e.g., $\sum_{i\in\mathcal{N}^-} x_{ij}^k = 1$), the number of people in the vehicle Q_j^k must be less than its capacity c_k. The value of Q_j^k is set to zero if vehicle k is not assigned to serve the demand at node j, which is achieved also by constraints (3.12). The constraints (3.13) and (3.14) together imply that for a pair of nodes successively visited by the same vehicle, the number of people travelling on the outbound arc from node j must be equal to the number of people on the arc from node i to j plus (or take away, if $q_j < 0$) the number of riders picked up (or dropped off, if $q_j < 0$) at node j. Note that constraints (3.13) or (3.14) are inactive when the nodes are not visited successively by a vehicle. Constraints (3.15) and (3.16) define that the number of people in each vehicle to be zero when it starts and arrives at the depots 0 and $\hat{n}$. Constraints (3.17) define the feasible sets of all decision variables.

We now explain the objective function (3.1). Define by

$$\mathcal{L}_i^k := \sum_{j\in\mathcal{N}} x_{ij}^k d_{ij} Q_i^k$$

the total travelling distance of all the people (including both the riders and the driver) transported by vehicle k after leaving node i and before arriving at the next node. Note that if node i is not visited by vehicle k we have $\mathcal{L}_i^k = 0$ (since $Q_i^k = 0$). Summing up $\mathcal{L}_i^k$ for all $i \in \mathcal{N}$ and $k \in \mathcal{V}$, we have

$$\mathcal{L} := \sum_{k\in\mathcal{V}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} x_{ij}^k d_{ij} Q_i^k .$$

Here we use $\mathcal{L}$ as a measure to the loading level of the whole ridesharing system.

In the same manner, we define the system's loading capability as follows.

$$\mathcal{D} := \sum_{k\in\mathcal{V}} c_k \left(\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} x_{ij}^k d_{ij} \right) .$$

Essentially this gives the total distance travelled if all vehicles were always fully loaded.

Our objective is to maximise the loading ratio of the whole system. The direct formulation $\mathcal{L}/\mathcal{D}$ leads to a nasty non-linear equation with quadratic terms in the numerator and variables in the denominator. Instead, we define by $\sigma \in (0,1)$ the target loading ratio and define the following objective function.

$$\max: \qquad \mathcal{L} - \sigma\mathcal{D}.$$

Substitute the definition of $\mathcal{L}$ and $\mathcal{D}$ and the objective function (3.1) is obtained. Despite still being a non-linear function, as we shall see in the next section, it can be reformulated into a linear form.

3.2 Reformulation

It is not difficult to see that $\mathcal{L}$ in the objective function (3.1) is a quadratic function. In this section we make a series of reformulations to simplify the problem to an integer programming problem. The first step aims at transforming the quadratic term in (3.1) into a linear form. We can reformulate the objective function as

$$\max_{\mathbf{x},\mathbf{Q}} \qquad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} c_k \left(\sum_{k\in\mathcal{V}} d_{ij}\min\left\{x_{ij}^k, c_k^{-1}Q_i^k\right\}\right) - \sigma\sum_{k\in\mathcal{V}} c_k\left(\sum_{i\in N}\sum_{j\in N} x_{ij}^k d_{ij}\right)$$

The equivalence between $c_k d_{ij}\min\{x_{ij}^k, c_k^{-1}Q_i^k\}$ and $d_{ij}x_{ij}^k Q_i^k$ is not difficult to be verified. If $x_{ij}^k = 0$ we have that

$$c_k d_{ij}\min\left\{x_{ij}^k, c_k^{-1}Q_i^k\right\} = x_{ij}^k d_{ij} Q_i^k = 0;$$

otherwise if $x_{ij}^k = 1$ we have that

$$c_k d_{ij}\min\left\{x_{ij}^k, c_k^{-1}Q_i^k\right\} = c_k d_{ij} c_k^{-1} Q_i^k = x_{ij}^k d_{ij} Q_i^k.$$

The first equation is from the fact that $Q_i^k \le c_k$ for all $i \in \mathcal{N}$ and $k \in \mathcal{V}$. By doing so, we can show that the objective function above is equivalent to (3.1).

Now we step further to reformulate problem (M1) into an integer program. To this end, we introduce an auxiliary variable τ_{ij}^k for all $k \in \mathcal{V}, i, j \in \mathcal{N}$. We have the following reformulation,

$$\max_{\mathbf{x},\mathbf{Q}} \quad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} c_k \sum_{k\in\mathcal{V}} d_{ij}\tau_{ij}^k - \sigma\sum_{k\in\mathcal{V}} c_k\left(\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} x_{ij}^k d_{ij}\right) \tag{3.18}$$

$$\tag{3.19}$$

$$\begin{aligned} \text{s.t.} \quad & (3.2) \sim (3.17), & \\ & \tau_{ij}^k \le x_{ij}^k & \forall i,j \in \mathcal{N}, \forall k \in \mathcal{V}, \qquad (3.20)\\ & \tau_{ij}^k \le c_k^{-1}Q_i^k & \forall i,j \in \mathcal{N}, \forall k \in \mathcal{V}, \qquad (3.21)\end{aligned}$$

The number of additional constraints introduced into this mixed integer program is $2 \times |\mathcal{N}| \times |\mathcal{N}| \times |\mathcal{V}|$.

4 Extensions

In this section, we extend our work in two directions. Firstly, we propose a flow-dependent model to include the congestions caused by pick-up and drop-off from the ridesharing activities. Secondly, we include the riders' individual evaluation on different transportation modes into a ridesharing model with choice behaviours.

4.1 Flow Dependency

Instead of the flow-independent model as (M1), we consider a flow-dependent scenario in this section and relax the assumption that the pick-up and drop-off events due to the ridesharing activities do not introduce additional costs (e.g. congestion, traffic delay, etc.) to the traffic system concerned. This allows us to extend our ridesharing model to include more realistic situations, such as the peak time period when drivers might be less willing to enter the locations with intensive ridesharing demand which may lead to pick-up and drop-off congestion and delays.

To our best knowledge, we make the first attempt to include the congestion into the ridesharing decisions, where the amount of delays depends on the flows on the arcs linking to the same node.

The underlying reason of the ridesharing congestion is caused by the limited parking/loading spaces in a certain area, which translates to a capacity constraint for ridesharing activities within this area. Different nodes in set $\mathcal{N}$ might locate geographically close and share the parking slots in the same area. The ignorance of the geographic relationship between nodes might lead to long queues for parking, and thus significantly impact the ridesharing solutions.

To include the geographic information into the model, we divide the entire road network into R areas indexed by $r = 1, 2, \cdots, R$, each of which contains a subset of nodes in $\mathcal{N}$ with geographical locations close to each other. Define a set of indicators

$$\Delta := \{\delta_{r,i},\ r = 1, 2, \cdots, R,\ i \in \mathcal{N}\}$$

to denote the relationship between nodes and the areas with $\delta_{r,i} = 1$ if node i geographically belongs to area r and otherwise $= 0$. According to the definition, the nodes with $\delta_{r,i} = 1$ for the same r are geographically close. The ridesharing activities occurring in these nodes might share the same parking or other limited traffic resources and thus cause delays in area r.

Define a ridesharing delay function $\varpi_r(\cdot)$ to measure the impedance of area r for different congestion levels of ridesharing activities. The value of the function varies with the total number of ridesharing activities occurring at area r. We use z_r to measure the total number of ridesharing activities occurring in area r. We have

$$z_r(\mathbf{x}) := \sum_{i \in \mathcal{N} \setminus (\mathcal{O}' \cup \mathcal{D}')} \delta_{r,i} \left(\sum_{i' \in \mathcal{N}} x_{i'i}^k + \sum_{j \in \mathcal{N}} x_{ij}^k \right),$$

which is the linear combination of the elements in $\mathbf{x}$. We require $\varpi_r(\cdot)$ to be non-decreasing and differentiable and $\varpi_r(\mathbf{x})$ to be convex with respect to x_{ij}^k for $i, j \in \mathcal{N}$.

In our model, we use the function proposed by the U.S. Bureau of Public Roads to characterize the queuing phenomena with limited service resources. There are various approaches to model the delays at a ridesharing node that can be seen as transfer nodes in logistics networks. Simulation and queueing models are two of them. Compared with simulation, queueing analysis is more prevalently used to derive analytical expressions and it is easy to be incorporated in tactical decision models. Quoting from Crainic [14], "... most time-related functions are built to reflect the increasingly larger delays that result when facilities of limited capacity must serve a growing volume of traffic. Such congestion functions are typically derived from engineering procedures and queueing models...". Petersen [42, 43] proposes several models for various components of the classification process and study models which are based on the physical characteristics of the terminals. Crainic et al. [15] identify that the precise data may be difficult to obtain and even if obtained, may not be necessary for planning at a tactical level. Thus they propose two analytical formulae to calculate classification delays based on $M/M/1$ queueing model. By the definition, we formulate

$$\varpi_r(\mathbf{x}) := \varpi_r^0 \left(1 + \alpha \left(\frac{z_r(\mathbf{x})}{C_r}\right)^{\gamma}\right)$$

where ϖ_r^0 is the service time when area r is not congested, C_r is the practical capacity, and $\alpha > 0$ and $\gamma > 0$ are tuning parameters.

We can view $\varpi_r(\mathbf{x})$ as the virtual 'service time' at a node within area r for pick-up or drop-off passengers. The constraints (3.10) on the arrival times can be reformulated as

$$s_i^k + \hat{\varpi}_j(\mathbf{x}) + t_{ij} - M\left(1 - x_{ij}^k\right) \le s_j^k, \quad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V},$$

where node j is in area r. Thus the virtual service time at node j is the same as $\varpi_r(\mathbf{x})$ and can be formulated as

$$\hat{\varpi}_j(\mathbf{x}) := \sum_{r=1}^{R} \delta_{r,i} \varpi_r \left(\sum_{i \in \mathcal{N} \setminus (\mathcal{O}' \cup \mathcal{D}')} \delta_{r,i} \left(\sum_{i' \in \mathcal{N}} x_{i'i}^k + \sum_{j \in \mathcal{N}} x_{ij}^k \right) \right).$$

Then the flow-dependent model can be formulated as

$$\begin{aligned}
\text{(M2)} \quad \max_{\mathbf{x},\mathbf{s},\mathbf{Q}} \quad & \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{V}} x_{ij}^k d_{ij} Q_i^k - \sigma \sum_{k \in \mathcal{V}} c_k \left(\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} x_{ij}^k d_{ij} \right) & (4.1)\\
\text{s.t.} \quad & (3.2) \sim (3.9), & \\
& s_i^k + \hat{\varpi}_j(\mathbf{x}) + t_{ij} - M\left(1 - x_{ij}^k\right) \le s_j^k, \quad \forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}, & (4.2)\\
& (3.11) \sim (3.17). &
\end{aligned}$$

Note that the difference between the flow-independent and flow-dependent models is the extra time used in completing ridesharing activities within the congested area. Compared to the former, the latter includes additional $N \times N \times V$ constraints.

4.2 Individual Evaluation

Traditionally, travel behaviour modelling has been based on the axioms of expected utility ([62] and [33]). Random utility based discrete-choice models (RUM) provide an econometric interpretation of expected utility theory. RUM has been developed considerably in the past three decades and specifically for route-choice modelling.

We now extend the model (M1) to the situation where the system may take into account the riders' individual evaluation when making the ride-matching recommendations. That is, the probability that the recommendation will be finally accepted by the corresponding rider depends on their preference over some other alternative transportation modes.

Assume that for each rider there exist V transportation modes indexed by $v = 0, 1, 2, \cdots, V$ with mode 0 being the ridesharing. In addition, we assume that the arrival time at the destination is the only factor influencing a rider's choice on these modes. Denote by θ_i^v the *transit time* between node $i \in \mathcal{O}'$ and $n+i \in \mathcal{D}'$ if for transportation mode v, and by $\tau_i^v := a_i + \theta_i^v$ the arrival time at the destination node $n+i$. For the ridesharing mode, we denote $\tau_i^0(\mathbf{s}) := \min_{n+i\in\mathcal{D}'} \left\{s_{n+i}^k,\ k = 1, 2, \cdots, K\right\}$ as the arrival time at the destination node $n+i$ following the system recommendation for $i \in \mathcal{O}'$, where $\tau_i^0(\bullet)$ is a function of decision variables $\mathbf{s} := \{s_{n+i}^k : n+i \in \mathcal{D}'\}$.

The transit time saved for a rider at node i to use a particular transportation mode v compared to the slowest one can be calculated by $\hat{\tau}_i(\mathbf{s}) - \tau_i^v$, where $\hat{\tau}_i(\mathbf{s}) = \max_v\{\tau_i^0(\mathbf{s}), \tau_i^v,\ v = 1, \cdots, V\}$. The logit choice probability for the rider at node i to choose mode v can be written as

$$P_i^v(\mathbf{s}) = \frac{\exp(\hat{\tau}_i(\mathbf{s}) - \tau_i^v)}{\exp\left(\hat{\tau}_i(\mathbf{s}) - \tau_i^0(\mathbf{s})\right) + \sum_{v=1}^V \left(\exp(\hat{\tau}_i(\mathbf{s}) - \tau_i^v\right)},$$

and thus $P_i^0(\mathbf{s})$ is the probability of choosing the ridesharing mode. Note that $P_i^0(\mathbf{s})$ can be rewritten as

$$P_i^0(\mathbf{s}) = \left(1 + \sum_{v=1}^V \exp\left(\tau_i^0(\mathbf{s}) - \tau_i^v\right)\right)^{-1}.$$

We now consider an optimisation model with the aim of maximizing the average probability that a rider in the system chooses the ridesharing mode, which is

$$\begin{array}{ll} \max\limits_{\mathbf{x},\mathbf{s},\mathbf{Q}} & \dfrac{1}{n}\displaystyle\sum_{i=1}^n \left(1 + \sum_{v=1}^V \exp\left(\tau_i^0(\mathbf{s}) - \tau_i^v\right)\right)^{-1} \\ \text{s.t.} & (3.2)\sim(3.17) \end{array}$$

However, it is more rational for the ridesharing system to maximise the overall efficiency with respect to a new constraint that requires that the probability of individual riders' acceptance of the ridesharing recommendations must not below a value $\beta \in (0, 1)$. We can write this constraint as

$$\inf_{i=1,2,\cdots,n} \left(1 + \sum_{v=1}^V \exp\left(\tau_i^0(\mathbf{s}) - \tau_i^v\right)\right)^{-1} \geq \beta,$$

which can be transformed into the following set of constraints

$$\left(1 + \sum_{v=1}^V \exp\left(\tau_i^0(\mathbf{s}) - \tau_i^v\right)\right)^{-1} \geq \beta, \quad \forall\, i = 1, 2, \cdots, n.$$

They are equivalent to the following convex constraints

$$\sum_{v=1}^{V} \exp\left(\tau_i^0(\mathbf{s}) - \tau_i^v\right) \le \beta^{-1} - 1, \quad \forall\, i = 1, 2, \cdots, n.$$

Thus we can write our optimisation model as

$$\begin{aligned} \text{(M3)} \quad \max_{\mathbf{x},\mathbf{s},\mathbf{Q}} \quad & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}} x_{ij}^k d_{ij} Q_i^k - \sigma \sum_{k\in\mathcal{V}} c_k \left(\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} x_{ij}^k d_{ij}\right) && (4.3)\\ \text{s.t.} \quad & (3.2) \sim (3.17) \\ & \sum_{v=1}^{V} \exp\left(\tau_i^0(\mathbf{s}) - \tau_i^v\right) \le \beta^{-1} - 1, \quad \forall\, i = 1, 2, \cdots, n. && (4.4) \end{aligned}$$

5 Algorithm

In this section, we investigate the solution algorithm under the framework of large neighbourhood search proposed by Shaw [58]. The main idea in the LNS is to gradually improve an existing solution by alternatively destroying and repairing them.

In our algorithm, we propose some new updating methods. For problem (M1), we define by *feasible set* all the solutions satisfying constraints (3.2∼3.17) and *current solution* the best solution so far yielded by the algorithm. Therefore, the current solution prescribes a sequence of nodes to be visited by each vehicle that achieves the maximum value of objective function among all the solutions so far. To find a new candidate, the algorithm performs two operations. Firstly, it *destroys* the current solution by removing some demands from the routes of selected drivers; and then *repairs* the solution by finding appropriate drivers and insert the unserved demands back into their routes. We will explain the details on how to perform the destroy and repair operations in the subsequent sections.

The motivation to use the LNS in our problem is its capability in searching a large scale of neighbourhood in each iteration. This implies that the LNS has the potential in returning a better local optimal solution and hence is more efficient than the other neighbourhood search methods. The LNS heuristic belongs to a class of heuristics known as *Very Large Scale Neighbourhood Search* (VLSN) algorithms [3]. In the LNS the neighbourhood is implicitly defined by the operations which are used to destroy and repair an incumbent solution [45]. When dealing with problems with a large number of constraints as in our model, the LNS can generate feasible neighbourhood of a solution easily, based on which the *insert* operation can also be easily performed with respect to the origin-destination pairs.

Algorithm 1: Framework for the LNS heuristic

```
    Define by $N_{\text{move}}$ the maximum number of remove operations in an iteration and define
    by $\{solutions\}$ the set of feasible solutions
1   Function LNS
2     generate initial solution $x$ from $\{solutions\}$, set $x_{optimal} = x$ and randomly generate
      $N$ from $\{1, 2, \cdots, N_{move}\}$
3     let $t = 0$ to index the iteration
4     repeat
5       remove $N$ ridesharing demands from current solution
6       assign unserved ridesharing demands to vehicles according to a matching level
7       sort the ridesharing demands assigned to and hence the nodes visited by each vehicle
8       insert sorted ridesharing demands to each route
9       generate $x'$
10      if neighbourhood acceptance-criterion met then
11        $x = x'$
12      if $f(x') > f(x)$ where $f$ is the objective function of the problem, then
13        $x_{\text{optimal}} = x'$
14      update the index of iteration $t = t + 1$
15    until stop-criterion met
16  return $x_{\text{optimal}}$
```

5.1 Initial Solution

The initial solution is generated by *parallel insertion*. We denote by r_k^t the route for vehicle k where t is the index of iteration in the LNS algorithm, which is set to 0 in the initialization step. In the algorithm we take r_k^t as an ordered sequence of arcs. If the nodes i and j are successively visited by vehicle k in route r_k^t, we have $x_{ij}^k = 1$, and $= 0$ otherwise. More specifically, we write a route r_k^t as a set of directed arcs and the order of them is fixed, for instance,

$$r_k^t := \{(0, k), \cdots, (m_j, m_{j+1}), \cdots, (m_l, m_{l+1}), \cdots, (m + 2n + k, \hat{n})\}.$$

We denote the *position* of arc (m_j, m_{j+1}) in route r_k^t being p if there exist $p - 1$ arcs before it in this route.

Definition 5.1 (Feasibility) *The following principles define a feasible route in the algorithm:*

a. for any t, arc $(0, k)$ and arc $(m + 2n + k, \hat{n})$ must be at the first and the last position in r_k^t;

b. for any t, if arcs (i, j) and (j', k) are at successive positions in route r_k^t, i.e.,

$$r_k^t = \{(0, k), \cdots, (i, j), (j', k), \cdots, (m + 2n + k, \hat{n})\},$$

we have $j = j'$;

c. for any t, if arc (i, j) is at a position p in route r_k^t for $j \in \mathcal{O}'$ and some $i \in \mathcal{N}$, then we have that there exists some $i' \in \mathcal{N}$ such that $(i', m + n + j)$ at position p' in route r_k^t with $p' > p$ where $m + n + j \in \mathcal{D}'$.

In the algorithm, we first construct m routes that only contain each driver's origin and destination, where we denote them by $r_k^0 := \{(0,k),\ (k, m+2n+k),\ (m+2n+k, \hat{n})\}$ for $k \in \{1,2,\cdots,m\}$ with the superscript $t=0$ being the index of the initial iteration. In addition, we define the length of route r_k^t by the position of its last arc, i.e., $(m+2n+k, \hat{n})$, and denote it by N_k^t.

Definition 5.2 (Insertion) *Given a route $r_k^t := \{(0,m_1),(m_1,m_2),(m_2,m_3),\cdots,(m_{N_k^t},\hat{n})\}$ with $m_1 = k$ and $m_{N_k^t} = m+2n+k$, the insertion of ridesharing demand at node i to r_k^t at positions j and l for $l > j$ is as follows*

$$r_k^t := \{(0,m_1),\cdots,(m_{j-1},i),(i,m_j),\cdots,(m_{l-1},m+2n+i),(m+2n+i,m_l),\cdots,(m_{N^t+k},\hat{n})\}$$

for $k \in \mathcal{V}$, $i \in \mathcal{O}'$, $m_j, m_{j+1}, m_l, m_{l+1}, \in \mathcal{N}$, $m+2n+i \in \mathcal{D}'$ and $m+2n+k \in \mathcal{D}$. We call j and l the positions *of the insertion.*

It is worth mentioning that in the remaining of the paper when the insertion positions are not specified, the ridesharing demand at node i is inserted into an appropriate position of route r_k^t that achieves the highest increment in the value of objective function (3.1).

It is not difficult to see that so far route r_k^0 for $k = 1,2,\cdots,m$ is a feasible solution to problem (M1). In our algorithm, we find some high quality initial solutions for a good start of the search procedure in the LNS. Since the quality of initial solutions is determined by how ridesharing demands are sequentially inserted into each route, we need to determine m sequences of ridesharing demands that are to be inserted into route r_k^0 for $k = 1,2,\cdots,m$.

More specifically, we introduce a new measure called *regret* to evaluate the potential ridesharing demands to be inserted into a route. Let Δf_{ir_k} denote the increment of the objective function value (i.e., regret) after inserting ridesharing demand i into route r_k. Set $\Delta f_{ir_k} = 0$ if the route is infeasible or the objective function value decreases after the insertion. Here we have removed the superscript t from the notation of r_k^t for convenience. The context will make it clear when the route is constructed in iteration t of the algorithm. Ridesharing demand at node $i \in \{m+1,\cdots,m+n\}$ is inserted to all the m routes tentatively and the regret in the value of the objective function (3.18) is recorded for each insertion, i.e., Δf_{ir_k} for $k = 1,2,\cdots,m$. These values are then sorted in descending order that results in a new sequence of $\Delta f_{ir_{i,l}}$, where $r_{i,l}$ denotes the route into which the insertion of the ridesharing demand at node i achieves the lth largest regret for $l \le m$. The inserted ridesharing demand that has the largest regret is selected by solving the following problem:

$$\max_{i \in \{m+1,\cdots,m+n\}} \sum_{l=2}^{m} \left(\Delta f_{i,r_{i,1}} - \Delta f_{i,r_{i,l}}\right).$$

5.2 Remove

Given route r_k^t, we denote by $\mathcal{R}_k^t$ the set of nodes visited by vehicle k in route r_k^t. For example, for route $r_k^t := \{(0,m_1),(m_1,m_2),(m_2,m_3),\cdots,(m_{N_k^t},\hat{n})\}$, we have $\mathcal{R}_k^t := \{0,m_1,m_2,m_3,\cdots,m_{N_k^t},\hat{n}\}$. In iteration $t = 1,2,\cdots$, the algorithm selects a number of ridesharing demands at the nodes from the set $\mathcal{P}_t := \left(\bigcup_{k=1}^m \mathcal{R}_k^t\right) \bigcap \mathcal{O}'$. Here we denote by N_t the number of ridesharing demands. After that, the algorithm removes them from corresponding route r_k^t for $k \in \mathcal{V}$. The ridesharing demands to be removed are selected from $\mathcal{P}_t$ with equal probabilities, and the origin and the destination nodes are removed from the route by the following definition of *remove* operation.

Definition 5.3 (Remove) *For route $r_k^t := \{(0,k), \cdots, (m_j, i), (i, m_{j+1}), \cdots, (m_l, m+2n+i), (m+2n+i, m_{l+1}), \cdots, (m+2n+k, \hat{n})\}$, if $i \in \mathcal{R}_k^t$, then the* remove *operation of ridesharing demand at node i from r_k^t results in a new route $\{(0,k), \cdots, (m_j, m_{j+1}), \cdots, (m_l, m_{l+1}), \cdots, (m+2n+k, \hat{n})\}$ for vehicle k.*

5.3 Matching Level

In this section, we propose a concept of *matching level* to measure how close a rider' itinerary is to the existing routes. This concept is used to insert an unsatisfied demand into an existing route of a particular vehicle.

In each iteration, we call the process of choosing a subset of unserved demands and inserting them into the corresponding routes r_k^t for $k \in \mathcal{V}$ as repair operation to the solution in iteration t. The unsatisfied demands include those removed from the other routes in the current solution, that is $\mathcal{R}_{k'}^t$ for $k' \in \mathcal{V}$ and $k' \neq k$, and those not visited in the current solution, i.e., the nodes in the set $\bar{\mathcal{P}}_t := \mathcal{O}' - \mathcal{O}' \cap \mathcal{P}_t$. To make the repair operation efficient, we design an evaluating scheme based on the matching level rather than the assignment with equal probabilities. In the rest of this section, we show how this approach can significantly enhance the efficiency of the algorithm, and how easy the matching level can be computed.

Example 1. Here we give an example on how to determine the matching level between two unserved ridesharing demands. Assume that the first demand's pick-up and destination nodes are i and $n+i$, and the second demand's pick-up and destination nodes are j and $n+j$. We define r_{ij} as the shortest feasible route connecting nodes $i, n+i, j$ and $n+j$ with its length denoted by $d(r_{ij})$. This means that r_{ij} is one of the following six routes and yields the shortest distance,

$$\begin{array}{ll} \{(i,j),(j,n+i),(n+i,n+j)\}, & \{(i,j),(j,n+j),(n+j,n+i)\}, \\ \{(i,n+i),(n+i,j),(j,n+j)\}, & \{(j,i),(i,n+j),(n+j,n+i)\}, \\ \{(j,i),(i,n+i),(n+i,n+j)\}, & \{(j,n+j),(n+j,i),(i,n+i)\}. \end{array}$$

Let $d_{i,n+i}$ represent the distance between pick-up node i and destination node $n+i$. Then the matching level between demands at nodes i and j is calculated by

$$\phi_{ij}^u := \frac{\max\{d_{ii'}, d_{jj'}\}}{d(r_{ij})}$$

and

$$\phi_{ij}^l := \frac{\min\{d_{ii'}, d_{jj'}\}}{d(r_{ij})}$$

where $i' = n+i$ and $j' = n+j$, and $\phi_{ij}^u, \phi_{ij}^l \in (0,1)$. In our algorithm, we define the matching level between ridesharing demands at node i and j as

$$\phi_{ij} = w^l \phi_{ij}^l + w^u \phi_{ij}^u \tag{5.1}$$

where w^l and w^u are the weights of ϕ_{ij}^l and ϕ_{ij}^u with $w^l + w^u = 1$. □

In any iteration of the heuristic, before inserting a new pair of origin and destination nodes into the route, we need to measure the impact when the same vehicle serves demands at nodes i and j. For a route in which nodes i and $n+i$ are already connected, the value of ϕ_{ij}^{l} is closer to 1, implying that it is likely to result in little change when inserting node j (and the corresponding destination node) into this route. This is also true when inserting node i into a route in which node j and the corresponding destination node are already connected. Figure 1 illustrates the above scenarios, where Figure 1(a) shows a case where ϕ_{ij}^{l} is close to 1 and Figure 1(b) shows a case where ϕ_{ij}^{l} is much less than 1. Therefore ϕ_{ij}^{l} is a reasonable measure to the matching level of the demands at nodes i and j. We

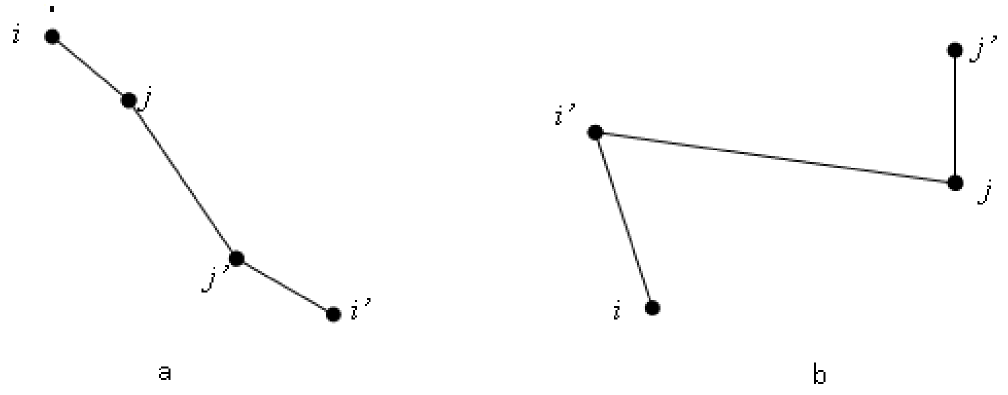

Figure 1: inserting nodes into a route with different matching level

extend the definition of matching level by taking ϕ_{ij}^{u} into account. When ϕ_{ij}^{u} is closer to 1, it is more likely to result in little change when removing node i or j and the corresponding destination nodes from a route with a direct link between them. Similarly, we can define the matching level between the nodes in $\mathcal{O}$ for the drivers and in $\mathcal{O}'$ for the riders, and the start and end node of vehicle k's itinerary is fixed at k and $m+2n+k$. The matching level between the demand at node i and vehicle k can be defined in the same way as in (5.1) by substituting j and j' by k and $k' = m+2n+k$ respectively. In addition, if there exist no feasible routes, the corresponding matching ratio is set to be $-\infty$.

We further extend the concept of matching level to characterize the efficiency of assigning a ridesharing demand to an existing route, which is defined as the average over the matching levels between this demand and all the other demands as well as the driver in this route. For example, for route $r_k^t := \{(0,k), \cdots, (m_j, i), (i, m_{j+1}), \cdots, (m_l, m+2n+i), (m+2n+i, m_{l+1}), \cdots, (m+2n+k, \hat{n})\}$, the matching level between ridesharing demand at node i and route r is

$$\phi_{ir} = \frac{\max\left\{\sum_{l \in \mathcal{O}' \cap \mathcal{R}_k^t} \phi_{il} + \phi_{ik}, 0\right\}}{n+1},$$

where $\mathcal{R}_k^t$ is the set of nodes visited by vehicle k. It is worth mentioning that if the matching level between the demand at node i and any node $j \in \mathcal{O}'$ served by route r or between the demand at node i and vehicle k is $-\infty$, it means that there is no feasible route and we define this matching level by 0.

The algorithm attains the priorities to the existing routes for inserting an unsatisfied demand as follows. Given the existing routes for m vehicles being $r_1, r_2, \cdots$, and r_m, the demand at node i is inserted to route r_l with probability:

$$P_{i,r_l} = \frac{\phi_{i,r_l}}{\sum_{j=1}^{m} \phi_{i,r_j}}, \tag{5.2}$$

which is essentially a variation of the roulette method proposed in [47].

Denote by l_k for all $k \in V$ the *matching threshold* for each route. The set of the demands to be added into vehicle k's route can be represented by $\mathcal{A}_k^t := \{i \,|\, P_{i,r_k^t} \geq \iota_k\}$. The demand at node i is added into vehicle k's route when $k = \arg\min_{k \in \mathcal{V}} P_{i,r_k^t}$. Note that if there exist more than one routes with the same P_{i,r_k^t}, the algorithm adds the demand at node i into a randomly selected route.

5.4 Optimal Insertion Positions

After determining how to insert a demand into each route, we now explain how to determine the position in each route to insert this demand.

Firstly, we define a *feasible position* P in route $r_k^t := \{(0, m_1), (m_1, m_2), (m_2, m_3), \cdots, (m_{N_k^t}, \hat{n})\}$ for the demand at node i if the arrival time of vehicle k at node i

$$s_i^k := s_0^k + \sum_{p=0}^{P} t_{m_p, m_{p+1}},$$

is within the time window $[a_i, b_i]$, and the number of people in vehicle k at node i

$$Q_i^k = \sum_{p=0}^{P} q_{m_p},$$

is within the capacity c_k of vehicle k.

We adapt the insertion heuristic based on the time difference method and search all the potential positions explicitly. The following parameters (as shown in the following table) are included in finding the insertion positions.

Table 2: Additional parameters

notation	
E_i	the earliest start time of pick-up at node $i \in \mathcal{O}'$;
L_i	the latest start time of pick-up at node $i \in \mathcal{O}'$;
Q_i	the number of people in a vehicle after operation at node i.

The values of E_i, L_i and Q_i are computed by a forward or backward recursion framework. For example, if nodes i and j are successively visited by the same vehicle, we can update these parameters by the following recursive framework

$$E_j = \max\{E_i + t_{ij}, a_j\}, \tag{5.3}$$
$$L_i = \min\{L_j - t_{ij}, b_i\}, \tag{5.4}$$
$$Q_j = Q_i + q_j. \tag{5.5}$$

Equation (5.3) implies that the earliest start time of pick-up at node j is the maximum between the earliest start time a_j and the earliest start time of the proceeding node i, i.e., E_i, plus the travelling time between these two nodes t_{ij}. In the same manner equation (5.4) determines the latest time of pick-up at node i, and (5.5) determines the number of people in the vehicle at successive nodes.

For origin node k and destination node $k' = m + 2n + k$, the following boundary conditions must be satisfied:

$$\begin{aligned} E_k &= a_k, \\ L_{k'} &= b_{k'}, \\ Q_k &= q_k. \end{aligned}$$

When node i' is inserted in between node i and j in a vehicle's route, the resulting new route is feasible if the following conditions hold

$$\begin{aligned} &\min\left\{L_j - t_{i'j}, b_{i'}\right\} \geq \max\left\{E_i + t_{ii'}, a_{i'}\right\}, \\ &L_j - E_i \geq t_{ii'} + t_{i'j}, \\ &Q_{i'} \leq c_k, \end{aligned}$$

where the first two inequalities are used to guarantee the relationship between the latest and earliest start times at these three nodes, while the third one is used to guarantee the number of people not more than the capacity of the vehicle. The details of the insertion operation is given in Algorithm 2.

Algorithm 2: Pseudocode for the insertion operation

```
1   compute $E_i, L_i, Q_i$ for each node $i$ in current route
2   for all positions in the route
3      if constraints (5.6), (5.6), (5.6) satisfied then
4         insert $p$ into current position, go to step 7
5      end if
6   repeat
7   update $E_i, L_i, Q_i$ for each node $i$
8   set the next position after $p$ as $I_{start}$
9   if $Q_i < c_k$ for all positions after $I_{start}$ then
10     set the last position as $I_{end}$
11  else set the first position that $Q_i > c_k$ as $I_{end}$
12  end if
13  for all positions between $I_{start}$ and $I_{end}$
14     if constraints (5.6), (5.6) satisfied then
15        compute the increase of objective function
16        if the increase is better than the best solution then
17           insert $p'$ to current position, set it as the best solution
18        end if
19     end if
20  repeat
```

5.5 Neighbourhood Acceptance Criterion

The solution of the insertion operation in Algorithm 2 is checked by a neighbourhood acceptance criterion. The main idea in the establishment of the neighbourhood acceptance criterion is to accept the candidate solution that can improve the objective function. We use this type of criteria to guarantee that the local optimality can be achieved in the LNS algorithm.

We propose a simulated annealing framework for the acceptance criterion. Let $\mathbf{x}$ and $\hat{\mathbf{x}}$ be the current solution and a solution in its neighbourhood respectively. The solution $\hat{\mathbf{x}}$ is accepted with probability

$$p(\hat{\mathbf{x}}, \mathbf{x}) = \min\left\{e^{f(\hat{\mathbf{x}})-f(\mathbf{x})/T_k}, 1\right\},$$

where $f(\mathbf{x})$ is the value of objective function in (3.1) for solution $\mathbf{x}$ and T_k is the temperature in the simulated annealing that decreases in each iteration by $T_k = T_0 c^k$. We set the initial temperature T_0 as:

$$T_0 = -\alpha\% \cdot |f(\mathbf{x})| \,/\, \ln 0.5. \tag{5.6}$$

Therefore a neighbourhood solution that is $\alpha\%$ worse than the current solution can still be accepted with a probability 50%. We reset the temperature to T_0 whenever a new solution is found.

We refer the readers to [66] for the details in simulated annealing. We also adapt a simulated annealing acceptance criterion in the insertion operation. By doing so, the situation where a feasible insertion is rejected because the objective function after several insertions can be improved. Considering that the change of objective function caused by insertion is relatively small, we set the temperature used in the insertion operation as $T' = T/m$.

5.6 Stopping Criteria

Three stopping criteria in the LNS algorithm are applied. First of all, we define a maximum number of iterations in the algorithm. Secondly, we consider a stopping criterion when the total execution time elapsed exceeds a given limit. The third stopping criterion is when the solution does not improve in a number of successive iterations.

6 Numerical Tests

In this section we test our proposed algorithm to a number of randomly generated instances, which are classified into two subsets, the small scale instances and a large scale instance. For the small instances, we use the computational time and the results of CPLEX as benchmarks to study the efficiency of our LNS algorithm. We show that our LNS algorithm can solve most of the small scale instances optimally. For the large scale instance, we decompose the problem into a number of smaller ones by clustering together the drivers and the ridesharing demands which are geographically close. CPLEX is then used to solve the problem for each sub-network, the optimal solutions to which are aggregated as an approximation to the solution of the original problem. To illustrate the efficiency of the LNS algorithm, we compare the approximation against the LNS solution by varying the number of subclasses. At the end of this section we study the impact of the congestion to the optimal solutions in a ridesharing system.

6.1 An Illustrative Example

We present an instance with 40 nodes and 20 people in total (both drivers and riders). The structure of the network is characterized by the following node-to-node matrix.

$$\begin{array}{c|cccccccccccccccccccccccccccccccccccccccc}
node & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 \\ \hline
1 & 0 & 3 & 1 & 6 & 3 & 3 & 3 & 3 & 5 & 3 & 3 & 3 & 1 & 1 & 1 & 3 & 6 & 4 & 1 & 3 & 1 & 3 & 3 & 3 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 4 \\
2 & 3 & 0 & 3 & 4 & 1 & 1 & 2 & 1 & 3 & 1 & 3 & 5 & 3 & 3 & 3 & 1 & 4 & 3 & 3 & 2 & 3 & 2 & 1 & 2 & 6 & 2 & 3 & 2 & 1 & 1 & 5 & 1 & 3 & 2 & 1 & 3 & 1 & 1 & 1 & 5 \\
3 & 1 & 3 & 0 & 6 & 3 & 3 & 3 & 3 & 5 & 3 & 3 & 3 & 1 & 1 & 1 & 3 & 6 & 4 & 1 & 3 & 1 & 3 & 3 & 3 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 4 \\
4 & 6 & 4 & 6 & 0 & 4 & 4 & 6 & 4 & 2 & 4 & 7 & 8 & 6 & 6 & 6 & 4 & 1 & 2 & 6 & 4 & 6 & 4 & 4 & 6 & 2 & 4 & 2 & 4 & 3 & 4 & 9 & 4 & 6 & 4 & 4 & 6 & 4 & 4 & 4 & 8 \\
5 & 3 & 1 & 3 & 4 & 0 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 0 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 5 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
6 & 3 & 1 & 3 & 4 & 1 & 0 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 5 & 1 & 3 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
7 & 3 & 2 & 3 & 6 & 3 & 3 & 0 & 3 & 4 & 3 & 2 & 5 & 2 & 2 & 2 & 3 & 6 & 5 & 3 & 3 & 3 & 3 & 2 & 1 & 7 & 3 & 5 & 2 & 3 & 3 & 4 & 2 & 3 & 3 & 3 & 2 & 3 & 3 & 3 & 3 \\
8 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 0 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 5 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
9 & 5 & 3 & 5 & 2 & 3 & 3 & 4 & 3 & 0 & 3 & 6 & 7 & 5 & 5 & 5 & 3 & 2 & 2 & 5 & 3 & 5 & 3 & 3 & 5 & 3 & 3 & 1 & 3 & 3 & 3 & 7 & 3 & 5 & 3 & 3 & 5 & 3 & 3 & 3 & 7 \\
10 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 0 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 6 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
11 & 3 & 3 & 3 & 7 & 4 & 4 & 2 & 4 & 6 & 4 & 0 & 4 & 2 & 2 & 2 & 4 & 7 & 6 & 2 & 4 & 2 & 4 & 4 & 2 & 9 & 4 & 6 & 3 & 4 & 4 & 2 & 3 & 2 & 4 & 4 & 2 & 4 & 4 & 4 & 2 \\
12 & 3 & 5 & 3 & 8 & 5 & 5 & 5 & 5 & 7 & 5 & 4 & 0 & 3 & 3 & 3 & 5 & 8 & 6 & 3 & 5 & 3 & 5 & 5 & 4 & 10 & 5 & 8 & 6 & 5 & 5 & 5 & 5 & 3 & 5 & 5 & 3 & 5 & 5 & 5 & 4 \\
13 & 1 & 3 & 1 & 6 & 3 & 3 & 2 & 3 & 5 & 3 & 2 & 3 & 0 & 1 & 1 & 3 & 6 & 5 & 1 & 3 & 1 & 3 & 3 & 2 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 3 \\
14 & 1 & 3 & 1 & 6 & 3 & 3 & 2 & 3 & 5 & 3 & 2 & 3 & 1 & 0 & 1 & 3 & 6 & 4 & 1 & 3 & 1 & 3 & 3 & 2 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 4 \\
15 & 1 & 3 & 1 & 6 & 3 & 3 & 2 & 3 & 5 & 3 & 2 & 3 & 1 & 1 & 0 & 3 & 6 & 5 & 1 & 3 & 1 & 3 & 3 & 2 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 3 \\
16 & 3 & 1 & 3 & 4 & 0 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 0 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 5 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
17 & 6 & 4 & 6 & 1 & 4 & 4 & 6 & 4 & 2 & 4 & 7 & 8 & 6 & 6 & 6 & 4 & 0 & 2 & 6 & 4 & 6 & 4 & 4 & 6 & 2 & 4 & 2 & 4 & 4 & 4 & 9 & 4 & 6 & 4 & 4 & 6 & 4 & 4 & 4 & 8 \\
18 & 4 & 3 & 4 & 2 & 2 & 2 & 5 & 2 & 2 & 2 & 6 & 6 & 5 & 4 & 5 & 2 & 2 & 0 & 5 & 2 & 5 & 2 & 3 & 5 & 4 & 2 & 3 & 4 & 2 & 2 & 8 & 3 & 5 & 2 & 2 & 5 & 2 & 2 & 2 & 7 \\
19 & 1 & 3 & 1 & 6 & 3 & 3 & 3 & 3 & 5 & 3 & 2 & 3 & 1 & 1 & 1 & 3 & 6 & 5 & 0 & 3 & 1 & 3 & 3 & 3 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 4 \\
20 & 3 & 2 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 0 & 3 & 1 & 2 & 3 & 6 & 1 & 4 & 3 & 2 & 1 & 6 & 2 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 6 \\
21 & 1 & 3 & 1 & 6 & 3 & 3 & 3 & 3 & 5 & 3 & 2 & 3 & 1 & 1 & 1 & 3 & 6 & 5 & 1 & 3 & 0 & 3 & 3 & 2 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 3 & 3 \\
22 & 3 & 2 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 0 & 2 & 3 & 6 & 1 & 4 & 3 & 2 & 1 & 6 & 2 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
23 & 3 & 1 & 3 & 4 & 1 & 1 & 2 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 3 & 3 & 2 & 3 & 2 & 0 & 3 & 5 & 2 & 3 & 2 & 1 & 1 & 5 & 1 & 3 & 2 & 1 & 3 & 1 & 1 & 2 & 5 \\
24 & 3 & 2 & 3 & 6 & 3 & 3 & 1 & 3 & 5 & 3 & 2 & 4 & 2 & 2 & 2 & 3 & 6 & 5 & 3 & 3 & 2 & 3 & 3 & 0 & 7 & 3 & 5 & 2 & 3 & 3 & 3 & 2 & 3 & 3 & 3 & 2 & 3 & 3 & 3 & 3 \\
25 & 8 & 6 & 8 & 2 & 5 & 5 & 7 & 5 & 3 & 6 & 9 & 10 & 8 & 8 & 8 & 5 & 2 & 4 & 8 & 6 & 8 & 6 & 5 & 7 & 0 & 6 & 3 & 6 & 5 & 6 & 10 & 6 & 8 & 6 & 6 & 8 & 5 & 6 & 6 & 10 \\
26 & 3 & 2 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 2 & 3 & 6 & 0 & 4 & 3 & 2 & 1 & 6 & 2 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
27 & 6 & 3 & 6 & 2 & 4 & 3 & 5 & 4 & 1 & 4 & 6 & 8 & 6 & 6 & 6 & 4 & 2 & 3 & 6 & 4 & 6 & 4 & 3 & 5 & 3 & 4 & 0 & 3 & 3 & 4 & 8 & 3 & 6 & 4 & 4 & 6 & 4 & 4 & 4 & 8 \\
28 & 4 & 2 & 4 & 4 & 2 & 2 & 2 & 2 & 3 & 2 & 3 & 6 & 4 & 4 & 4 & 2 & 4 & 4 & 4 & 3 & 4 & 3 & 2 & 2 & 6 & 3 & 3 & 0 & 2 & 3 & 5 & 2 & 4 & 3 & 2 & 4 & 2 & 2 & 3 & 5 \\
29 & 3 & 1 & 3 & 3 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 2 & 3 & 2 & 1 & 3 & 5 & 2 & 3 & 2 & 0 & 1 & 6 & 1 & 3 & 2 & 1 & 3 & 1 & 1 & 1 & 5 \\
30 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 6 & 1 & 4 & 3 & 1 & 0 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 & 5 \\
31 & 4 & 5 & 4 & 9 & 6 & 6 & 4 & 6 & 7 & 6 & 2 & 5 & 4 & 4 & 4 & 6 & 9 & 8 & 4 & 6 & 4 & 6 & 5 & 3 & 10 & 6 & 8 & 5 & 6 & 6 & 0 & 5 & 4 & 6 & 6 & 4 & 6 & 6 & 6 & 2 \\
32 & 3 & 1 & 3 & 4 & 1 & 1 & 2 & 1 & 3 & 1 & 3 & 5 & 3 & 3 & 3 & 1 & 4 & 3 & 3 & 2 & 3 & 2 & 1 & 2 & 6 & 2 & 3 & 2 & 1 & 1 & 5 & 0 & 3 & 2 & 1 & 3 & 1 & 1 & 1 & 5 \\
33 & 1 & 3 & 1 & 6 & 3 & 3 & 3 & 3 & 5 & 3 & 2 & 3 & 1 & 1 & 1 & 3 & 6 & 5 & 1 & 3 & 1 & 3 & 3 & 3 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 0 & 3 & 3 & 1 & 3 & 3 & 3 & 3 \\
34 & 3 & 2 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 2 & 3 & 6 & 1 & 4 & 3 & 2 & 1 & 6 & 2 & 3 & 0 & 1 & 3 & 1 & 1 & 1 & 6 \\
35 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 6 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 0 & 3 & 1 & 1 & 1 & 5 \\
36 & 1 & 3 & 1 & 6 & 3 & 3 & 2 & 3 & 5 & 3 & 2 & 3 & 1 & 1 & 1 & 3 & 6 & 5 & 1 & 3 & 1 & 3 & 3 & 2 & 8 & 3 & 6 & 4 & 3 & 3 & 4 & 3 & 1 & 3 & 3 & 0 & 3 & 3 & 3 & 3 \\
37 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 5 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 0 & 1 & 1 & 5 \\
38 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 1 & 3 & 6 & 1 & 4 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 0 & 1 & 5 \\
39 & 3 & 1 & 3 & 4 & 1 & 1 & 3 & 1 & 3 & 1 & 4 & 5 & 3 & 3 & 3 & 1 & 4 & 2 & 3 & 1 & 3 & 1 & 2 & 3 & 6 & 1 & 4 & 3 & 1 & 1 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 0 & 5 \\
40 & 4 & 5 & 4 & 8 & 5 & 5 & 3 & 5 & 7 & 5 & 2 & 4 & 3 & 4 & 3 & 5 & 8 & 7 & 4 & 6 & 3 & 5 & 5 & 3 & 10 & 5 & 8 & 5 & 5 & 5 & 2 & 5 & 3 & 6 & 5 & 3 & 5 & 5 & 5 & 0
\end{array}$$

In addition, the time windows $[a_i, b_i]$ and the number of riders to be picked up (q_i) at node i is specified in the matrix below.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
a_i	0	1	0	0	0	0	0	0	6	2	6	3	7	0	2	7	9	5	6	7	0	0	0	0	0	0	0	2	7	0	3	8	3	6	8	5	7	2	3	5
b_i	1	7	4	14	7	5	8	8	10	9	9	11	13	6	6	14	11	9	10	13	8	16	15	16	17	14	10	7	9	3	6	9	11	8	12	10	16	8	10	12
q_i	2	1	7	6	2	4	7	6	2	4	1	3	1	6	7	1	5	5	2	3	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

Without loss of generality, we also consider the time windows for the destinations of drivers and riders, and thus characterize their preferences on the terminal times for their itineraries. Note that $i = 21$ to 40 are the nodes of the destinations of the drivers and the riders. Besides we have $\sigma = 0.2$ and the number of vehicles being 7. The solution from the LNS algorithm is shown in Table 3. It is shown that in the optimal solution five vehicles are used for picking up the ridesharing demands in the network. Note that the demand at node 20 with the destination node of 33 cannot be satisfied in the optimal solution. In Figure 2, we plot the objective function value obtained in each iteration of the algorithm. It shows that these values converge within 16 iterations.

6.2 Small Scale Instances

For convenience, we name the instances by "Number-of-Nodes σ" where 'A'∼'G' are used to denote σ that takes a value between 10% and 70% by an interval of 10%. For instance, the instance 40A

Table 3: The routes of vehicles for the illustrative example.

vehicle no.	route
1	1→34;
2	2→8→15→21→28→35;
3	3→12→19→25→32→36;
4	4→18→31→37;
5	5→17→30→38;
6	6→14→27→39;
7	7→40.

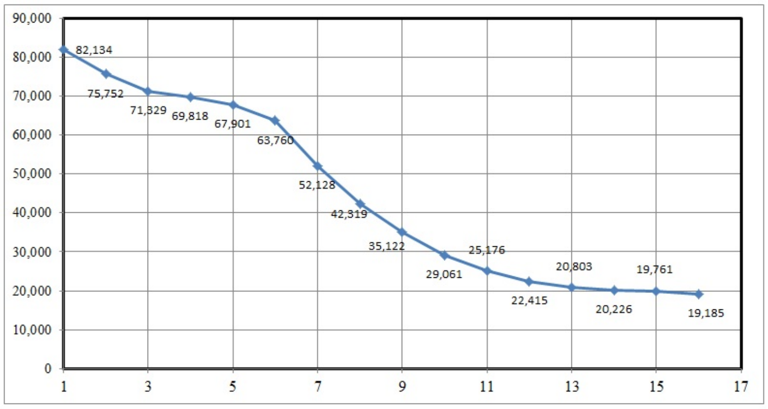

Figure 2: Objective values of the LNS algorithm for the illustrative example.

means that there are 40 nodes in the traffic network and the loading ratio σ is 10%. The nodes in these small instances are fully connected and the geographical information of the nodes in these instances are randomly generated.

The results on these small scale instances are presented in Table 4. It was run on a Core i3 PC with 4G memory. For the LNS algorithm, we set the iteration limit by $Num = 3000$. In addition, we set α in (5.6) and $T_k = 0.9987$ of simulated annealing mechanism in the neighbourhood acceptance criterion. The results in Table 6 show that the LNS heuristic can solve all the small scale instances in reasonable times. For a subset of these problems with comparatively large scales, we cannot get an exact solution by CPLEX, such as, $60A$, $60C$, $70A \sim C$. For instances $50A$ and $50B$, CPLEX consumes extraordinarily computational time, while the LNS algorithm can return acceptable results (less than 2% to the optimal values) in significantly shorter time.

6.3 The Large Scale Instance

We divide the large scale instance into several smaller ones and solve them independently by CPLEX. Firstly, we use k-means method to cluster the vehicles into S groups according attributes such as origins, destinations and time windows. We perform the clustering several times, choosing S vehicles randomly as the initial clustering centres each time, and select the one with the minimal

Table 4: Results of small scale instances.

Instance	σ	CPLEX		LNS		Difference	
		Obj.	CPU(s)	Obj.	CPU(s)	Obj. gap	Route
40A	0.1	35521	23	35521	13	0	-
40B	0.2	19185	11	19185	17	0	-
40C	0.3	-5399	3	-5399	15	0	-
40D	0.4	-13430	3	-13430	12	0	-
40E	0.5	-37141	6	-37141	12	0	-
40F	0.6	-35253	2	-35253	11	0	-
40G	0.7	-56556	2	-56556	13	0	-
50A	0.1	37176	2498	36471	19	1.90%	20/30
50B	0.2	22903	1590	22833	18	0.31%	4/36
50C	0.3	-2201	4	-2201	11	0	-
50D	0.4	-24563	12	-24563	12	0	-
50E	0.5	-37854	3	-37854	10	0	-
50F	0.6	-49649	3	-49649	11	0	-
50G	0.7	-44764	2	-44764	10	0	-
60A	0.1	-	-	51043	26	-	-
60B	0.2	1707	100	1707	20	0	-
60C	0.3	-	-	1856	18	0	-
60D	0.4	-13077	17	-13077	17	0	-
60E	0.5	-35704	10	-35704	34	0	-
60F	0.6	-27744	5	-27744	19	0	-
60G	0.7	-64607	5	-64607	18	0	-
70A	0.1	-	-	35569	26	-	-
70B	0.2	-	-	17598	24	-	-
70C	0.3	-	-	-9264	20	-	-
70D	0.4	-22882	9	-22882	21	0	-
70E	0.5	-43789	4	-43789	24	0	-
70F	0.6	-57067	6	-57067	23	0	-
70G	0.7	-72580	6	-72580	37	0	-

average distance as the result of clustering.

Then ridesharing demands are assigned to each vehicle according to the increased route length caused by such assignment. If the demand at node i is served by vehicle k, the increased route length ϕ'_{ik} can be calculated as:

$$\phi'_{ik} = d_{ki} + d_{ii'} + d_{ik'} - d_{kk'}$$

where i' is the drop-off node for the rider and the k' is the destination of the driver. The ridesharing demand at node i is assigned to vehicle k with the minimal value of ϕ'_{ik}.

If we cluster the vehicles into less subclasses, each subclass will have more ridesharing demands. It is more likely that CPLEX will fail to find feasible solutions. So we start with a large number of subclasses and reduce the number of them gradually until CPLEX fails. We restrict the maximum number of ridesharing demands in each subclass to be less than 1.3 times of the average demands of all subclasses. We further restrict that the nodes contained in the subclass with just one vehicle should be less than half of the average nodes across all subclasses.

The instance concerned contains 50 vehicles, 250 ridesharing demands, and 600 nodes. We set $Num = 1000$, $N_{\text{move}} = 5$, $\alpha = 0.05$, and $c = 0.9987$. The test was run on a Core i7 PC with 8G memory. The results are reported in Table 5 and Figure 3. Clearly, LNS heuristic outperforms CPLEX in the quality of solution and computing time. Note that the dotted line in Figure 3 is a polynomial approximation of objective function value with respect to the number of clusters.

Table 5: Results of the large scale instance.

	No. of clusters	Obj.	CPU(s)
CPLEX	19	46738	112
	18	48152	135
	17	42988	283
	16	51485	681
	15	52718	1776
	14	64803	3218
LNS	-	86183	1503

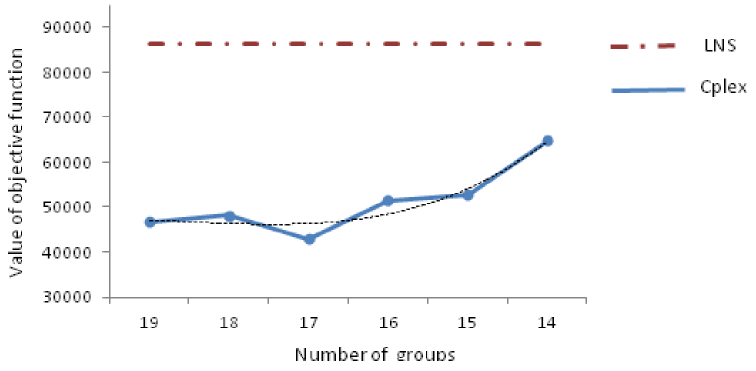

Figure 3: Comparison of LNS and CPLEX for the large instance.

6.4 Extensions

6.4.1 Instances with Congestions

In this section we present some numerical tests to show how the congestion generated by pick-up and drop-off activities affect the optimal value of the objective function. The LNS algorithm was applied to solve both model M1 and M2 for each of the instances.

We reconsider a set of instances investigated in Section 6.2 (from 40A to 50G). In these instances, we assume that there are two geographically clustered areas for the start nodes and the destination nodes, respectively. Therefore we have $R = 4$ in model (M2). In addition, we randomly group the start/destination nodes into two subsets. In other words the values of $\delta_{r,i}$ are randomly generated in $\{0, 1\}$ for $r = 1, \cdots, 4$ and all $i \in \mathcal{N}$. In our tests, for every r we use a simple form of function $\varpi_r(\mathbf{x})$ where we let $\varpi_r^0 = 1$, $\alpha = 0.3$, $\gamma = 1$ and C_r being $10 \times N_r$ with N_r being the number of nodes in area r. It is worth mentioning that we have let $\gamma = 1$ to simplify model (M2) so that all its constraints are linear. This may not be the case for many real world problems, but such a treatment allows us to focus the analysis on the impacts from ridesharing congestions.

In Table 6, we compare the results between M2 and M1 for every instance to illustrate how the congestion affects the optimal values of the objective function. We observe that the optimal values have been significantly reduced when the congestion caused by ridesharing activities is taken into consideration. In addition, since we assume there are two sets for the start and destinations nodes respectively, the instances with 50 nodes are affected more significantly than those with 40 nodes. From the computational aspect, model M2 takes longer CPU times than its counterpart M1 for problems with more nodes.

Table 6: Results of model M1 and M2.

Instance	σ	M2		M1	
		Obj.	CPU(s)	Obj.	CPU(s)
40A	0.1	31960	13	35521	13
40B	0.2	13046	20	19185	17
40C	0.3	-12606	19	-5399	15
40D	0.4	-25706	14	-13430	12
40E	0.5	-52487	16	-37141	12
40F	0.6	-53668	13	-35253	11
40G	0.7	-78041	12	-56556	13
50C	0.3	-4329	24	-2201	11
50D	0.4	-28819	22	-24563	12
50E	0.5	-44238	23	-37854	10
50F	0.6	-58161	26	-49649	11
50G	0.7	-54508	22	-44764	10

6.4.2 Instances with Individual Evaluations

In this section we present some numerical tests to illustrate the influence of riders' individual evaluations on the routing results of vehicles and the optimal value of the objective functions. The LNS algorithm was applied to solve both model M1 and M3 for each of the instance.

We investigate again a set of instances in Section 6.2 (from 40A to 50G). In these instances, we define the value of $\beta = 0.9$ uniformly for all the riders. We assume that there are two geographically clustered areas for the start nodes and the destination nodes, respectively. Therefore we have $R = 4$ in model M3. In addition, we randomly group the start/destination nodes into two subsets. We simply assume that there are other two transportation modes, i.e., $v = 1, 2$. In the example, we consider the case where $\tau_i^v := \alpha_v d_i + \epsilon_{i,v}$ where d_i is the time used to travel between the OD pair of rider i. The values of coefficients α_1 and α_2 are 1.50 and 0.75 respectively. Moreover, $\epsilon_{i,v}$ is the random variables characterizing the uncertainties in traveling times between the OD pair of rider i when using the transportation mode v. In the example, we let ϵ_v following a normal distribution with mean at 0 and standard deviation at $5\% \times \alpha_v d_i$.

We perform model M3 for cases 40A to 50G. In Table 7, we compare the results between M3 and M1 for every instance to illustrate how the riders' individual evaluations affect the optimal values of the objective function. We observe that the optimal values have been significantly reduced when the riders' individual evaluations are taken into account. The results in Table 7 show that model M3 takes much longer CPU times than the counterparts M1, which is particularly true due to the nonlinear constraints in M3.

Table 7: Results of model M1 and M3.

Instance	σ	M2		M1	
		Obj.	CPU(s)	Obj.	CPU(s)
40A	0.1	30437	329	35521	13
40B	0.2	11869	331	19185	17
40C	0.3	-14612	383	-5399	15
40D	0.4	-30006	423	-13430	12
40E	0.5	-64112	412	-37141	12
40F	0.6	-74881	439	-35253	11
40G	0.7	-90331	494	-56556	13
50C	0.3	-12812	477	-2201	11
50D	0.4	-39114	531	-24563	12
50E	0.5	-68327	523	-37854	10
50F	0.6	-87323	588	-49649	11
50G	0.7	-80811	602	-44764	10

7 Conclusions

In this paper, we have considered a ride-matching and routing problem with an objective to maximise the loading ratio of the whole ridesharing system, which takes a non-linear form. We have transformed the objective function into a linear equation and formulated the problem as an integer program. We then made two extensions to the model. In the first one the congestion caused by pick-up and drop-off activities in ridesharing systems is considered, while in the other one the riders' individual evaluation on alternative transportation modes is taken into account. To solve the resulting models, which are of large sizes for practical problems, we have developed a large neighbourhood search algorithm. The search procedure is significantly accelerated by evaluating the matching level between unsatisfied ridesharing demands and existing routes. Numerical tests on a number of randomly generated instances show that the LNS heuristic can solve small scale instances optimally and return satisfactory solutions for large scale instances quickly.

Acknowledgements The authors are grateful to two anonymous referees and the Editor-in-Chief for their helpful comments and suggestions.

References

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

[2] N. A. H. Agatz, A. Erera, M. W. P. Savelsbergh and X. Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 2012, 233: 295–303.

[3] R.K. Ahuja, O. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very largescale neighbourhood search techniques.*Discrete Applied Mathematics*, 2002, 123: 75-102.

[4] J. R. Araque, G. Kudva, T. L. Morin and J. F. Pekny. A branch and cut algorithm for the vehicle routing problem. *Annal of Operations Research*, 1994, 50, 37-59.

[5] R. Baldacci, V. Maniezzo and A. Mingozzi, An exact method for the car pooling problem based on Lagrangean column generation. *Operations Research*, 2004, 52: 422–439.

[6] S. R. Balseiro, I. Loiseau and J. Ramonet. An Ant Colony Algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows. *Computers and Operations Research*, 2011, 38: 954–966.

[7] R. Bent and P. Van Hentenryck. A two-stage hybrid algorithm for pick-up and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 2006, 33(4): 875–893.

[8] B. Catay. *Ant colony optimisation and its application to the vehicle routing problem with pick-ups and deliveries*, in: Chiong, R., Dhakal, S. (Eds.), Natural Intelligence for Scheduling, Planning and Packing Problems. Springer Berlin Heidelberg, 2009, 219–244.

[9] N. D. Chan and S. A. Shaheen. Ridesharing in North America: past, present, and future. Transport Reviews 32 (1), 2012, 93-112.

[10] M. Cherkesly, G. Desaulniers and G. Laporte. Branch-price-and-cut algorithms for the pick-up and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, 2014.

[11] M. Cherkesly, G. Desaulniers and G. Laporte. A population-based metaheuristic for the pick-up and delivery problem with time windows and lifo loading. *Computers & Operations Research,* 2015, 62: 23–35.

[12] J. F. Cordeau, A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 2006, 54, 573–586.

[13] J. F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 2003, 37: 579-594.

[14] T. G. Crainic, Long-Haul Freight Transportation., *Long-Haul Freight Transportation. s.l.:Kluwer*, pp.451-516.

[15] T. G. Crainic, J. Ferland, A. Rousseau, A tactical planning model for rail freight, *Transportation Science*, 18, pp.165-184, 1984.

[16] J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 1986, 6: 301–325.

[17] M. Diana and M. M. Dessouky. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, 2004, 38: 539–557.

[18] Y. Dumas, J. Desrosiers and F. Soumis. The pick-up and delivery problem with time windows. *European Journal of Operational Research*, 1991, 54(1): 7–22.

[19] T. Gaynor, 2015. *Solving the Critical Mass Problem. What is the Simplest Thing that Could Work?* Accessed 7/29/2015.

[20] M. Gendreau, F. Guertin, J.-Y. Potvin and R. Sguin, Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technology* , 2006, 14(3): 157-174.

[21] A. Fabri and P. Recht, On dynamic pick-up and delivery vehicle routing with several time windows and waiting times. *Transportation Research Part B: Methodological*, 2006, 40(4): 335-350.

[22] M. Furuhata, M. M. Dessouky, F. Ordonez, M. Brunet, X. Wang and S. Koenig. Ridesharing: the state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 2013, 57: 28–46.

[23] R. W. Hall and A. Qureshi. Dynamic ride-sharing: Theory and practice. *Journal of Transportation Engineering*, 1997, 123(4): 308–315.

[24] D. Hartmann. Google challenge winners give android thumbs-up. *VentureBeat*, 2008.

[25] J. J. Jaw, A. R. Odoni, H. N. Psaraftis and N. H. M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, 1986, 20: 243–257.

[26] G. Laporte, Fifty years of vehicle routing, *Transportation Science*, 2009, 43(4): 408–416.

[27] P. Larranaga, C. M. H. Kuijpers, R. H. Murge, I. Inza and S. Dizdarevic. Genetic algorithms for the Travelling Salesman Problem: A review of representations and operators. *Artificial Intelligence Review*, 1999, 13: 129–170.

[28] H. C. Lau, Z. Liang. Pickup and delivery with time windows: Algorithms and test case generation. *International Journal on Artificial Intelligence Tools*, 2002, 11(03): 455–472.

[29] A. N. Letchford, R. W. Eglese and J. Lysgaard. Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 2002, 94 21-40.

[30] A. Levofsky and A. Greenberg. Organized dynamic ride sharing: The Potential environmental benefits and the opportunity for advancing the concept, Paper No. 01- 0577, *Transportation Research Board, 2001 Annual Meeting, Washington, DC, January*, 2001.

[31] H. Li and A. Lim, A metaheuristic for the pick-up and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 2003, 12(02): 173–186.

[32] Q. Lu and M. M. Dessouky, An exact algorithm for the multiple vehicle pick-up and delivery problem. *Transportation Science*, 2004, 38: 503–514.

[33] R. D. Luce and H. Raiffa, *Games and Decisions,* John Wiley & Sons, New York, 1957.

[34] O. B. G. Madsen, H. F. Ravn and J. M. Rygaard. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities and multiple objectives. *Annal of Operations Research*, 1996, 60: 193-208.

[35] O. Madsen, H. Ravn, and J. Rygaard, A heuristic algorithm for a dial-aride problem with time windows, multiple capacities, and multiple objectives. *Annal of Operations Research*, 1995, 60(1):193208.

[36] N. Masoud, R. Lloret-Batlle and R. Jayakrishnan, Using bilateral trading to increase ridership and user permanence in ridesharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 2017, 102: 60–77.

[37] S. Mitrović-Minić and G. Laporte, Waiting strategies for the dynamic pick-up and delivery problem with time windows. *Transportation Research Part B: Methodological*, 2004, 38(7): 635-655.

[38] W. P. Nanry and J. W. Barnes. Solving the pick-up and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 2000, 34(2): 107–121.

[39] S. N. Parragh, K. F. Doerner and R. F. Hartl. A Survey on Pickup and Delivery Problems. Part II: Transportation between pick-up and delivery locations. *Journal fur Betriebswirtschaft*, 2008, 58:81–117.

[40] S. N. Parragh, K. F. Doerner and R. F. Hartl. Variable neighbourhood search for the dial-a-ride problem.*Computers and Operations Research*, 2010, 37: 1129–1138.

[41] D. Pelzer, J. Xiao, D.Zehe, M. H. Lees, and A. C. Knoll, A partition-based match making algorithm for dynamic ridesharing, *IEEE Transactions on Intelligent Transportation Systems*, 16(5): 2015, 2587–2598.

[42] E. R. Petersen, Railyard modelling - Part I: Prediction of putthrough time,*Transportation Science*,Vol.11,pp.50-59.

[43] E. R. Petersen, Railyard modelling - Part II: The effect of yard facilities on congestion, *Transportation Sciences*,Vol.11,pp.50-59.

[44] V. Pillac, M. Gendreau, C. Guéret, and A. Medaglia, A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 2013, 225(1): 1-11.

[45] D. Pisinger and S. Ropke, Large neighbourhood search. *Handbook of Metaheuristics, the series International Series in Operations Research & Management Science*, 2010, 146: 399–419.

[46] J. Potvin and J. Rousseau. *Constraint-directed search for the advanced request dial-a-ride problem with service quality constraints*, Computer Science and Operations Research: New Developments in Their Interfaces. Pergamon Press, Oxford, 457–474, 1992.

[47] C. Prines, A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 2004, 31: 1985–2002.

[48] H. N. Psaraftis, A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 1980, 14(2): 130–154.

[49] H. N. Psaraftis, An exact algorithm for the single vehicle many-to-many dial-A-ride problem with time windows. *Transportation Science*, 1983, 17: 351–357.

[50] S. Raney, San francisco to silicon valley, california, instant ridesharing with transfer hub, Transport. Res. Rec.: J. Transport. Res. Board (2143), 134-141, 2010.

[51] S. Ropke, J. F. Cordeau and G. Laporte. Models and a branch-and-cut algorithm for pick-up and delivery problems with time windows. *Networks*, 2007, 49(4): 258–272.

[52] S. Ropke and J. F. Cordeau. Branch and cut and price for the pick-up and delivery problem with time windows, *Transportation Science*, 2009, 43: 267–286.

[53] S. Ropke and D. Pisinger. An adaptive large neighbourhood search heuristic for the pick-up and delivery problem with time windows. *Transportation science*, 2006, 40(4): 455–472.

[54] D. Sáez, C. E. Cortés, and A. Núñez, Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computer and Operations Research* , 2008, 35(11): 3412-3438.

[55] M. Sahin, G. Cavuslar and T. Öncan. An efficient heuristic for the multi-vehicle one-to-one pick-up and delivery problem with split loads. *Transportation Research Part C: Emerging Technologies*, 2013, 27: 169–188.

[56] M. W. P. Savelsberg and M. Sol. The general pick-up and delivery problem, *Transportatoin Science*, 1995, 29: 17–29.

[57] M. W. P. Savelsberg and M. Sol. Drive: Dynamic routing of independent vehicles, *Operations Research*, 1998, 46: 474–490.

[58] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In CP-98 (*Fourth International Conference on Principles and Practice of Constraint Programming*), *Lecture Notes in Computer Science*, 1998, 1520: 417-431.

[59] M. M. Solomon and J. Desrosiers. Time window constrained routing and scheduling problems, , *Transportation Science*, 1988, 22: 1–13.

[60] P. Toth and D. Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 1997, 31: 60-71.

[61] T. Vidal, T. Crainic, M. Gendreau, and C. Prins, Heuristics for multiattribute vehicle routing problems: A survey and synthesis, *European Journal of Operational Research*, 2013, 231(1): 1-21.

[62] J. Von-Neumann and O. Morgenstern, *Theory of Games and Economic Behavior,* Princeton University Press, Princeton, 1944.

[63] X. Wang, M. Dessouky and F. Ordonez. A pick-up and delivery problem for ridesharing considering congestion. *Transportation Letters*, published-online.

[64] S. Winter and S. Nittel, Ad-hoc shared-ride trip planning by mobile geosensor networks. *International Journal of Geographic Information Science*, 00 (00), 1 21, 2006.

[65] X. Xing, T. Warden, T. Nicolai and O. Herzog, Smize: a spontaneous ride-sharing system for individual urban transit. *In: Proceedings of the 7th German Conference on Multiagent System Technologies, MATES09. Springer-Verlag, Berlin Heidelberg,* pp. 165176, 2009.

[66] V. F. Yu and S-Y Lin. A simulated annealing heuristic for the open location-routing problem. *Computers and Operations Research*, 2015, 62: 184–196.