Computers & Industrial Engineering 149 (2020) 106601

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Ride-sharing under travel time uncertainty: Robust optimization and clustering approaches

Yinglei Li[a], Sung Hoon Chung[b,*]

[a] Breakthrough Fuel, LLC, United States
[b] Department of Systems Science and Industrial Engineering, Binghamton University, United States

ARTICLE INFO

Keywords:
Ride-sharing
Clustering
Robust optimization
Uncertain travel time

ABSTRACT

In this paper, a ride-sharing system that supports frequent updates of participants' information is studied, for which driver-rider matching and associated routes need to be decided quickly. Uncertain travel time is considered explicitly when matching and route decisions are made; a robust optimization approach is proposed to handle it properly. To achieve computational tractability, an extended insertion algorithm in conjunction with a tabu search method is proposed, and a cluster-first-route-second approach is used to find heuristic solutions. In particular, a greedy heuristic and *k*-means algorithm are used to group the riders and their respective results, along with non-clustering case outcomes, are compared. Numerical examples show that the ride-sharing system considered in this paper can be a viable solution by means of our proposed approaches even for challenging cases where the scale of the system is large and decisions need to be made quickly.

1. Introduction

Ride-sharing is a transportation mode where travelers having similar itineraries share a vehicle to, in general, reduce costs. The notion of dynamic ride-sharing is introduced to address the system where participants' itineraries and schedules vary frequently and their information is provided in a very short notice (Agatz, Erera, Savelsbergh, & Wang, 2012). Most dynamic ride-sharing research aims to find the best matches between riders and drivers, and provide a shortest route for a driver (Agatz et al., 2012; Alonso-Mora, Samaranayake, Wallar, Frazzoli, & Rus, 2017; Furuhata et al., 2013; Nourinejad & Roorda, 2016; Pelzer et al., 2015; Stiglic, Agatz, Savelsbergh, & Gradisar, 2016; Wang, Agatz, & Erera, 2017). In this paper, a ride-sharing problem is studied where a quick match needs to be repeatedly found between drivers and riders. Although the proposed model in this paper is static, it aims to help achieve the goal of dynamic ride-sharing by proposing a computationally-tractable clustering approach for instantaneous matching and providing best routes considering the travel time uncertainty explicitly by means of robust optimization, as they are a prerequisite to a success of the dynamic ride-sharing.

Ride-sharing has attracted researchers' attention not only because of its economical benefits, but also thanks to its positive environmental and societal impacts such as reducing air pollution, traffic congestions, etc (Agatz et al., 2012; Alonso-Mora et al., 2017; Chan & Shaheen, 2012; Ferguson, 1997; Furuhata et al., 2013; Kelly, 2007; Morency, 2007; Nourinejad & Roorda, 2016; Pelzer et al., 2015; Stiglic et al., 2016; Wang et al., 2017). The wide-spread adoption of ride-sharing, however, has been limited despite its benefits, which is mainly due to its disorganized coordination, safety concerns, and lack of effective methods to encourage participation (Furuhata et al., 2013). The coordination has been particularly difficult for ride-sharing that requires an immediate matching while participants are consistently changing.

Ride-sharing is not a new idea but it has gained a resurgence of interest over the last decade thanks to the recent development of information technology that may help the coordination of ride-sharing in a variety of ways including smart phone apps and social media services (Agatz et al., 2012). However, there are still barriers for ride-sharing to overcome for its wide-adoption. One is the lack of computationally efficient algorithms that can provide a real-time match for drivers (servers) and riders (clients) and its associated route (Agatz et al., 2012). Consequently, there has been a growing need to address the computation issues, especially for ride-sharing that requires immediate matching upon request. The development of computationally tractable methods and algorithms for real life-size ride-sharing problems is one of the key topics in the current research. Travel time uncertainty is another barrier to the ride-sharing problem. In most ride-sharing problems, a driver may visit several different locations to pick up riders on the way to a final destination and, therefore, the chance that the driver

* Corresponding author.
E-mail address: schung@binghamton.edu (S.H. Chung).

https://doi.org/10.1016/j.cie.2020.106601
Received 10 September 2019; Received in revised form 22 May 2020; Accepted 6 June 2020
Available online 27 June 2020
0360-8352/

will be subject to uncertain travel time gets higher compared to the non ride-sharing case. As participants have time constraints in most cases, the consideration of uncertain travel time plays a significant role in coordinating ride-sharing. In particular, travel time uncertainty is greatly related to the effectiveness of driver-rider matching. According to the recent study about the role of participants' flexibility to matching (Stiglic et al., 2016), the willingness of participants to depart from the origin and arrive at the destination a little earlier or later, respectively (the flexibility of time window), is the main factor influencing the success rate of matching. In addition, the willingness of drivers and/or riders to detour to pick up additional riders (flexible maximum driving/riding time) also influence the rate of matches. Therefore, the explicit consideration of travel time uncertainty is a must as it may significantly impact flexibility of time windows and detours, which in turn very closely related to matching, one of the key issues for successful ride-sharing.

In this paper, a ride-sharing problem is studied where drivers (servers) and riders (clients) may have different origins and destinations. The riders' requests and the available drivers information are collected and stored in a centralized system. Not only a one-to-one match between a driver and a single rider but also a one-to-multiple match (up to the capacity of a vehicle) is considered. Given all the drivers and riders information in the same pool (e.g., same service area), multiple matches (each consisting of one driver and one or more riders) are made at the same time to minimize the overall travel cost while the matching rate is maximized. Such information is assumed to be updated continuously via electronic bulletin in social media or some smart phone applications. In this setting, the goal is to increase the matching rate and provide a robust route quickly for each driver under travel time uncertainty. Note that the mathematical model for ride-sharing proposed in this paper is not dynamic but static, the purpose of which is to provide a computationally tractable means for solving life-size problems quickly to find instantaneous matches and robust routes upon request. The proposed static model can be solved repeatedly as the new request arrives using a certain time interval, e.g., every 2 min.

Several constraints must be considered. First, drivers may have their own preferred time to depart from their homes/origins. Even if they are willing to spend some time to pick up riders, it is reasonable to assume that there exists an upper limit on the total travel time drivers are willing to spend. Likewise, riders may have their preferred departure time from their homes/origins as well. In addition, drivers and riders need to arrive at the destinations before their respective deadlines. Furthermore, vehicles have limited capacity. Drivers' routes need to be determined based on those constraints. Note that some riders may not be picked up due to limited number of drivers. Travel time uncertainty may have a significant impact on ride-sharing, as most constraints mentioned above involve time as a primary factor. In some unfortunate cases, a feasible route satisfying all the constraints may become infeasible if the time between pick-up locations deviates from its nominal value. To overcome such an issue, a robust optimization approach is proposed.

In this paper, a vehicle routing problem (VRP), which is *NP-hard*, is used as a basis for the ride-sharing problem. Therefore, it is impractical to find an exact optimal route for large scale problems. Indeed, computational tractability is of particular interest in this paper because a robust route needs to be found as quickly as possible using continuously updated information; otherwise it will not be viable. For computational tractability, a hybrid heuristic algorithm based on an insertion algorithm and tabu search is considered first. To this end, the insertion algorithm proposed by Campbell and Savelsbergh (2004) is extended to consider capacity constraints, maximum travel time constraints, and riders that are left unserved while finding initial feasible routes. Then, tabu search is employed to improve the solution iteratively. To improve the tabu search based method, a cluster-first-route-second approach is proposed where drivers and riders are divided into several groups such that each group consists of one driver and several riders. Naturally, this decomposed problem can be solved much faster. Two clustering approaches: greedy heuristic and k-means clustering are used. In the greedy heuristic, the closest rider from a driver is assigned to the driver. Then, the next closest rider is included in the group. This iteration stops when the group size reaches the vehicle capacity. In k-means clustering, riders are partitioned into k groups based on their locations while minimizing the within-cluster summation of distances.

The remainder of this paper is structured as follows. a literature review is presented in Section 2. Model formulation and algorithms are shown in Sections 3 and 4. A numerical study is conducted and shown in Section 5 and Section 6 concludes the paper.

2. Literature review

Ride-sharing problems have attracted scholars in the filed of transportation and operations research, and the review of ride-sharing problems has been provided (Agatz et al., 2012; Chan & Shaheen, 2012; Furuhata et al., 2013). A dynamic ride-sharing problem finds riders with similar itineraries and time schedules on short notice and aims to provide a best route for a driver who is willing to pick up the riders (Agatz et al., 2012). Features of such dynamic ride-sharing problem include consistently changing driver and rider information, non-recurring trips, and prearranged routes. That is, the ride-sharing should be established (prearranged) on short notice that can range from a few minutes to a few hours before departure time (dynamic driver and rider information), which will be a single (non-recurring) trip. These features distinguish dynamic ride-sharing from traditional carpooling or vanpooling, both of which require a long-term commitment among two or more people to travel together on recurring trips for a particular purpose.

A number of studies have been conducted for various aspects of dynamic ride-sharing including matching between drivers and riders, routing for the drivers, and pricing. For some recent examples regarding the matching issues, a partition-based matching algorithm for dynamic ride-sharing is proposed (Pelzer et al., 2015); a centralized matching model is proposed for one-to-one match between driver and passenger and a decentralized matching model is tested based on agent based simulation (Nourinejad & Roorda, 2016); and the impact of different types of participants' flexibility on the performance of a single-driver, single-rider ride-sharing system is studied in which results indicate that small increases in flexibility may significantly increase the expected matching rate (Stiglic et al., 2016). More recently, the notion of stable or nearly-stable matches is proposed and then studied by several mathematical programming methods (Wang et al., 2017). In terms of routing, the ride-sharing with a vehicle of unlimited capacity, which is realistic in some situations (e.g., number of riders is fixed and less than the vehicle capacity), is considered to address the question if this assumption can relieve the computational burden for finding routes (Fanelli & Greco, 2015). In addition, the New York City taxi data is utilized to consider the real-time high-capacity ride-sharing that scales to large numbers of passengers and trips (Alonso-Mora et al., 2017). To dynamically generate optimal routes with respect to online demand and vehicle locations, the algorithm proposed by Alonso-Mora et al. (2017) starts from a greedy assignment and improves through a constrained optimization, quickly generating solutions of good quality and converging to the optimal assignment over time. With respect to pricing, a discounted trade reduction pricing mechanism for ride-sharing with dynamic demand is proposed (Zhang, Wen, & Zeng, 2016) and an online pricing mechanism for autonomous mobility-on-demand systems is studied (Shen, Lopes, & Crandall, 2016). Furthermore, a generalized ride-sharing system that allow users to schedule multi-modal trips in a single task is developed (Cangialosi, Di Febbraro, & Sacco, 2016).

Researchers have proposed various solution methodologies for ride-sharing and its variants, which can be roughly divided into the two: exact and heuristic methods. In the former, the vast majority of cases are large-scale methods such as column generation, branch-cut-and-

price, decomposition, etc. For example, an exact method for a carpooling problem based on Lagrangian column generation is proposed (Baldacci, Maniezzo, & Mingozzi, 2004); a branch-and-price method for a reliability oriented dial-a-ride model is suggested (Deleplanque, Quilliot, & Bernay, 2014); and a Lagrangian decomposition approach is proposed to solve the shared-taxi problem formulated as a mixed integer program (Hosni, Naoum-Sawaya, & Artail, 2014). The branch-and-bound and kinetic tree methods are compared (Huang, Bastani, Jin, & Wang, 2014) to solve the ride-sharing problem in which the results indicate that the kinetic tree algorithm outperforms the other. A mixed integer programming model to minimize the total travel time is formulated for which linearization and symmetry breaking methods are used to solve the model (Armant & Brown, 2014) and a coalition formulation algorithm and bounding technique are proposed for ride-sharing problems (Bistaffa, Farinelli, & Ramchurn, 2014). In terms of recent heuristic methods, a greedy algorithm and local search to solve the problem of online stochastic ride-sharing and taxi sharing is used (Manna & Prestwich, 2014); a greedy randomized adaptive search procedure is proposed to solve a dynamic ride-sharing problem with the objective of maximizing the number of served requests and minimizing the total cost (Santos & Xavier, 2015); and a variable neighborhood descent method is employed to minimize the total travel time is dynamic ride-sharing (Pinson, Afsar, & Prodhon, 2016).

For the examples of information processing and communication support systems as they are the requisite to the success of ride-sharing and carpooling, the use of a distributed geographic information system (GIS) with recent information and communication technologies (Calvo, de Luigi, Haastrup, & Maniezzo, 2004), mobile geosensor networks (Winter & Nittel, 2006), and traffic information grid (Fu, Fang, Jiang, & Cheng, 2008) are considered and employed to provide information for dynamic ride-sharing community service. Furthermore, hierarchical cloud architecture is used (Lin & Shen, 2016) to develop a wireless social network aided vehicle sharing system called VShare. When a user send a travel request to VShare, it will identify the carpooler in nearby locations. If no carpool is found within nearby locations, the travel requests will be matched by the hierarchical cloud server architecture.

In the literature, cluster-first-route-second algorithms have been used for vehicle routing problems and their variants, one of which is a capacitated location-routing problem (CLRP) that aims to minimize routing and location costs at the same time. In CLRP, the locations of depots, the set of customers assigned to each depot, and the optimal distribution routes are to be decided. Several hierarchical and non-hierarchical clustering techniques (with several proximity functions) are integrated with a sequential heuristic algorithm (Barreto, Ferreira, Paixao, & Santos, 2007) and a greedy clustering method and an ant colony algorithm are used (Mehrjerdi & Nadizadeh, 2013) to solve the CLRP with fuzzy demand problem. For multi-depot vehicle routing problems, a three-phase heuristic/algorithmic approach is considered (Dondo et al., 2007) where a time-window based heuristic algorithm is used to group the customer nodes into feasible clusters and a new type of genetic clustering algorithm, which is based on geometric shapes, is proposed (Yücenur & Demirel, 2011).

Although the clustering approach is widely used in vehicle routing problems, there are relatively limited number of studies employing clustering methods in the ride-sharing problem. A sequential clustering and routing approach is used to solve a dial-a-ride problem (Jorgensen, Larsen, & Bergvinsdottir, 2007), where a genetic algorithm is used to cluster customers into several groups and then a routing problem for each cluster is solved. A genetic algorithm based approach to perform both the clustering and routing tasks at the same time is then proposed (Herbawi & Weber, 2012).

The contribution of this paper is summarized as follows: (1) a new formulation is developed to address different origins and destinations for drivers and riders, (2) uncertain travel time is explicitly considered, and (3) a new tractable heuristic method is proposed.

3. Ride-sharing model and algorithms

In this section, we propose a ride-sharing model to find the optimal matches between riders and drivers and at the same time find the optimal routes for drivers in which multiple drivers and multiple riders are considered. In addition, a hybrid algorithm based on insertion algorithm and tabu search is proposed to solve this model.

3.1. Deterministic model for ride-sharing

The ride-sharing problem considered in this paper contains several common assumptions (Agatz et al., 2012): (1) The ride-sharing is arranged on short-notice (a few minutes or within one hour before the earliest departure time of participants); (2) The ride-sharing is for non-recurring trips, which means that drivers and riders are different in each single arrangement; (3) The ride-sharing is pre-arranged based on participants' constraints, and only the trips that satisfy the drivers' and riders' constraints are conducted; (4) There must be at least one driver and one rider; and (5) The objective of ride-sharing is to minimize the participants' total travel cost by utilizing the vehicle's available space while not sacrificing their convenience too much. One of the ways to realize the objective is by matching a driver with riders having similar origins and destinations as the driver's. This is to minimize the amount of detours and maximize the length of shared route. A driver picks up riders at their origins up to the vehicle capacity and drops them off at their destinations. It is assumed that riders are not allowed to get off from the vehicle before the all the pick-ups are done, and drivers are not allowed to pick up riders once the drop-offs start. This implies that a rider cannot get off before another rider gets on. This is not only for the sake of simplicity for also consistent with other assumptions such as pre-arranged ride-sharing that is commonly found in the literature (Agatz, Erera, Savelsbergh, & Wang, 2011; Agatz et al., 2012).

The notation used in the mathematical model is as follows. Let K be the set of drivers and R be the set of riders. In each arrangement of ride-sharing, $|K| \geqslant 1$and $|R| \geqslant 1$. The origin and destination of driver $k \in K$are denoted by $b_k \in V_s$ and $w_k \in V_s'$ where V_s and V_s' represent the set of the drivers' origins and destinations, respectively. The origin–destination (o-d) pair of driver k is then denoted by (b_k, w_k) and the set of the drivers' origin–destination pairs is denoted by P_s. Similarly, the origin, destination, and pair of them of rider r are denoted by $b_r \in V_c$, $w_r \in V_c'$, and $(b_r, w_r) \in P_c$ where V_c, V_c', and P_c represent the set of the riders' origins, destinations, and o-d pairs, respectively. Let a_{kr} denote the decision variable to determine whether driver k will serve rider r. Driver k will transport the rider r from the origin of rider r to the destination of rider r when $a_{kr} = 1$, otherwise, $a_{kr} = 0$. Let $V = V_s \cup V_s' \cup V_c \cup V_c'$ and $i \in V$ be an index for a node (location) that is defined as either an origin or a destination for drivers and/or riders. Let $V_r = V_c' \cup V_c \setminus \{b_r\}$ denote the set of the riders' origins and destinations excluding b_r and $V_r' = V_c \cup V_c' \setminus \{w_r\}$ denote the set of the riders' origins and destinations excluding w_r. The transportation network considered in this problem is a complete network and $(i, j) \in A$ denotes an arc in the transportation network representing the shortest feasible path between the nodes i and j, where A represent the set of arcs in the complete network. In addition, let t_{ij} be the travel time between nodes i and j, and x_{ij}^k, $(i, j) \in A$, $k \in K$ be a binary variable that is equal to 1 only if arc $(i, j)$is traversed by driver k. Here, the assumption is that the travel cost is proportional to the travel time, and the coefficient to convert travel time to travel cost is denoted by f. In addition, let y_r, $r \in R$, be a binary variable that is equal to 1 if the rider r is not served by any

Table 1
Notation summary.

Symbols	Description	Symbols	Description
K	Set of drivers, $k \in K$	R	Set of riders, $r \in R$
b_k	Origin of driver k	w_k	Destination of driver k
V_s	Set of the drivers' origins	V_s'	Set of the drivers' destinations
(b_k, w_k)	Origin–destination (o-d) pair of driver k	P_s	Set of the drivers' origin–destination pairs
b_r	Origin of rider r	w_r	Destination of rider r
V_c	Set of the riders' origins	V_c'	Set of the riders' destinations
(b_r, w_r)	Origin–destination (o-d) pair of rider r	P_c	Set of the riders' origin–destination pairs
a_{kr}	Decision variable to determine whether driver k will serve rider r	i, j	indices for nodes (geographical locus)
V_r	$V_r = V_c' \cup V_c \setminus \{b_r\}$, set of the riders' origins and destinations excluding b_r	V_r'	$V_r' = V_c \cup V_c' \setminus \{w_r\}$, set of the riders' origins and destinations excluding w_r
(i, j)	Arc in the transportation network representing the shortest path between the nodes i and j	A	Set of arcs in the network
t_{ij}	Travel time between locations i and j	x_{ij}^k	Binary variable that is equal to 1 only if arc (i, j) is traversed by driver k
f	Coefficient to convert travel time into travel cost	y_r	Binary variable that is equal to 1 if the rider r is not served by any drivers
p_r	Associated penalty if the pick-up request of rider r is not served by any driver	s_i	Nonnegative variable representing the arrival time at node i
e_{b_k}	Earliest time that driver k is willing to depart from his origin b_k	l_{b_k}	Latest time that driver k is willing to depart from his origin b_k
e_{w_k}	Earliest time that driver k is willing to arrive at his destination w_k	l_{w_k}	Latest time that driver k is willing to arrive at his destination w_k
e_{b_r}	Earliest time that rider r is willing to depart from his origin b_r	l_{b_r}	Latest time that rider r is willing to depart from his origin b_r
e_{w_r}	Earliest time that rider r is willing to arrive at his destination w_r	l_{w_r}	Latest time that rider r is willing to arrive at his destination w_r
q_k	Number of seats available for driver k	m_k	Maximum requests that driver k is willing to serve
t_k	Maximum driving time driver k is willing to spend	d_r	Amount of demand in the request by rider r
M	A large number that is used in the constraints to eliminate the sub-tours		

drivers and p_r is the associated penalty if the pick-up request of rider r is not served by any driver. Moreover, s_i is a nonnegative variable representing the arrival time at node i. Let e_{b_k} and l_{b_k} denote the earliest time and the latest time that driver k is willing to depart from his origin b_k, respectively. Likewise, let e_{w_k} and l_{w_k} denote the earliest time and latest time that driver k is willing to arrive at his/her destination. For rider $r \in R$, we can define e_{b_r}, l_{b_r}, e_{w_r}, and l_{w_r} in the same fashion. Let q_k denote the number of seats available for driver k;m_k denote the maximum requests that driver k is willing to serve; t_k denote the maximum driving time driver k is willing to spend; and d_r be the amount of demand (the number of people to be picked up) in the pick-up request of rider r as there may be more than one rider in the same request (e.g., family, friends, etc). Let M denote a large number that is used in the constraints to eliminate the sub-tours, $M \geqslant t_k$, $\forall\, k \in K$. The notation is summarized in Table 1.

The ride-sharing model can be formulated as a mixed-integer program as follows:

$$\min \sum_{k\in K}\sum_{(i,j)\in A} ft_{ij}x_{ij}^k + \sum_{r\in R} p_r y_r \tag{1}$$

$$\text{s. t.} \sum_{k\in K} a_{kr} + y_r = 1 \;\forall\, r \in R \tag{2}$$

$$\sum_{r\in R} d_r a_{kr} \leqslant q_k \;\forall\, k \in K \tag{3}$$

$$\sum_{r\in R} a_{kr} \leqslant m_k \;\forall\, k \in K \tag{4}$$

$$\sum_{j\in\{b_k\}\cup V_r\setminus\{w_r\}} x_{jb_r}^k - a_{kr} = 0 \;\forall\, k \in K, \forall\, r \in R \tag{5}$$

$$\sum_{j\in V_r} x_{b_r j}^k - a_{kr} = 0 \;\forall\, k \in K, \forall\, r \in R \tag{6}$$

$$\sum_{j\in V_r'} x_{jw_r}^k - a_{kr} = 0 \;\forall\, k \in K, \forall\, r \in R \tag{7}$$

$$\sum_{j\in\{w_k\}\cup V_r'\setminus\{b_r\}} x_{w_r j}^k - a_{kr} = 0 \;\forall\, k \in K, \forall\, r \in R \tag{8}$$

$$\sum_{j\in\{w_k\}\cup V_c} x_{b_k j}^k = 1 \;\forall\, k \in K \tag{9}$$

$$\sum_{j\in\{b_k\}\cup V_c'} x_{jw_k}^k = 1 \;\forall\, k \in K \tag{10}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij}^k \leqslant t_k \;\forall\, k \in K \tag{11}$$

$$s_j - s_i \geqslant t_{ij} - M\left(1 - \sum_{k\in K} x_{ij}^k\right) \;\forall\, (i,j) \in A \tag{12}$$

$$t_{b_k i}x_{b_k i}^k - s_i \leqslant 0 \;\forall\, k \in K, \forall\, i \in V_c \cup \{w_k\} \tag{13}$$

$$s_{b_r} - e_{b_r}(1 - y_r) \geqslant 0 \;\forall\, r \in R \tag{14}$$

$$s_{b_r} - l_{b_r}(1 - y_r) \leqslant 0 \;\forall\, r \in R \tag{15}$$

$$s_{w_r} - e_{w_r}(1 - y_r) \geqslant 0 \;\forall\, r \in R \tag{16}$$

$$s_{w_r} - l_{w_r}(1 - y_r) \leqslant 0 \;\forall\, r \in R \tag{17}$$

$$e_{b_k} \leqslant s_{b_k} \leqslant l_{b_k} \;\forall\, k \in K \tag{18}$$

$$e_{w_k} \leqslant s_{w_k} \leqslant l_{w_k} \;\forall\, k \in K \tag{19}$$

$$s_{w_k} \geqslant s_{w_r} + t_{w_r,w_k} - M(1 - a_{kr}) \;\forall\, r \in R, \forall\, k \in K \tag{20}$$

$$x_{ij}^k \in \{0, 1\} \;\forall\, (i,j) \in A, \forall\, k \in K \tag{21}$$

$$a_{kr} \in \{0, 1\} \;\forall\, k \in K, \forall\, r \in R \tag{22}$$

$$y_r \in \{0, 1\} \ \forall \ r \in R \tag{23}$$

$$s_{b_r}, s_{w_r}, s_{b_k}, s_{w_k} \geqslant 0 \ \forall \ r \in R, \forall \ k \in K \tag{24}$$

The objective function (1) minimizes the total cost of the participants in the ride-sharing, including the total travel cost of the drivers and the cost due to the penalties of unserved riders. Constraints (2) ensure that rider r is served by at most one driver and the variable $y_r$will be 1 if rider r is not served by any drivers. Constraints (3) ensure that the capacity constraints of the vehicles are satisfied during the ride-sharing. Constraints (4) ensure that the number of requests assigned to driver k does not exceed the maximum requests that driver k is willing to serve. Constraints (5) and (6) ensure that driver k needs to visit the origin of rider r if rider r is assigned to driver k. Constraints (7) and (8) ensure that driver k needs to visit the destination of rider r if rider r is assigned to driver k. Constraints (9) and (10) ensure that driver k leaves from his origin and arrives at his destination. Constraints (11) are the maximum time constraints. Constraints (12) and (13) ensure the sub-tours are eliminated and the arrival times are presented correctly. Constraints (14) and (15) ensure that rider r will be picked up at his origin within his given time window if rider r is matched with a driver. Constraints (16) and (17) ensure that rider r will arrive at his destination within his given time window if rider r is matched with a driver. Constraints (18) ensure that driver k will depart from his origin within his given time window. Constraints (19) ensure that driver k will arrive at his destination within his given time window. Constraints (20) ensure that driver k will send his assigned riders to their destinations before arriving at his destination.

Note that (1) only considers driver's travel time/cost but not riders' travel/time cost. This is mainly because one of the primary objectives is to minimize the number of unserved riders, and it is assumed that riders' travel is feasible as long as their departure/arrive time requirements are met. However, if riders' travel time is indeed a concern, the objective function (1) and constraints can be modified to include riders' travel time/cost. This will require the use of demand variable d_r in the objective function and the addition of variables, x_{ij}^r, to denote if arc (i, j) is traversed by driver r.

If a participant does not have specific earliest departure time or earliest arrival time, the earliest departure time or earliest arrival time for this participant are set as the time when he/she posts the ride-sharing request. If a participant does not have specific latest departure time or latest arrival time, the latest departure time or latest arrival time for this participant are set as infinity. In addition, it is assumed that the rider r is ready to depart and picked up from his/her origin within the given time window, which implies that that the driver k does not need to wait for rider r as long as the driver's arrival time at the origin of rider r is within rider r's time window.

As the time window flexibility plays a very important role for matching (Stiglic et al., 2016), a driver's waiting time at node i, denoted by δ_i, may be added to the model by modifying Constraints (12) as follows:

$$s_j - s_i \geqslant t_{ij} + \delta_i - M\left(1 - \sum_{k \in K} x_{ij}^k\right) \ \forall \ (i, j) \in A \tag{25}$$

Note that the above ride-sharing problem is in the formalism of mixed integer programming, which can be solved using off-the-shelf software such as CPLEX and Gurobi if the size of the problem is small enough. However, our ride-sharing problem is based on vehicle routing formulation, which is known to be*NP-hard*, such that heuristics is required as the size of problem gets larger. We present the extended insertion algorithm and tabu search method accordingly in the subsequent section.

3.2. Robust counterpart

Travel time uncertainty can play a critical role in our ride-sharing problem, as the optimal solution could be even infeasible (some time window constraints may be violated) when realized travel time is significantly different than its nominal value. In this section, let us consider how to find a robust solution against the travel time uncertainty.

The realized travel time t_{ij}, $(i, j) \in A$ is assumed to be in the range $[\bar{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}]$, where $\bar{t}_{ij}$ is the nominal value of travel time and $\hat{t}_{ij}$ is the maximum delay. Note that while this range would be defined as $[\bar{t}_{ij} - \hat{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}]$in many robust-related studies, the one-sided range $[\bar{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}]$is much more common in robust vehicle routing problems (Ordónez, 2010) because arriving early to the destination has no harm to participants while using one-sided uncertainty box set can contribute to computational tractability. Let J^k be the set of arcs subject to travel time uncertainty and let Γ_k be the parameter to control the budget of uncertainty (Ordónez, 2010). Also, let S^k be the subset of J^k and its size is controlled by Γ_k. Now, it is ready to formulate the robust counterpart of the ride-sharing problem:

$$\min \ f \sum_{k \in K} \left(\sum_{(i,j) \in A} \bar{t}_{ij} x_{ij}^k + \max_{\{S^k | S^k \subseteq J^k, |S^k| \leqslant \Gamma_k\}} \sum_{(i,j) \in S^k} \hat{t}_{ij} x_{ij}^k \right) + \sum_{r \in R} p_r y_r \tag{26}$$

$$\text{s. t.} \quad (2) - (10), (14) - (24)$$

$$\sum_{(i,j) \in A} \bar{t}_{ij} x_{ij}^k + \max_{\{S^k | S^k \subseteq J^k, |S^k| \leqslant \Gamma_k\}} \sum_{(i,j) \in S^k} \hat{t}_{ij} x_{ij}^k \leqslant t_k \forall \ k \in K \tag{27}$$

$$s_j - s_i \geqslant (\bar{t}_{ij} + \hat{t}_{ij}) - M\left(1 - \sum_{k \in K} x_{ij}^k\right) \ \forall \ (i, j) \in A \tag{28}$$

$$(\bar{t}_{b_k i} + \hat{t}_{b_k i}) x_{b_k i}^k - s_i \leqslant 0 \ \forall \ k \in K, i \in V_c \cup \{w_k\} \tag{29}$$

Here, Γ_k decides the level of robustness. If $\Gamma_k = 0$, then the robust counterpart is the same as the original deterministic model. If Γ_k is set to be the maximum possible value, then it will be the worst case, i.e., all arcs are set to experience their maximum delays. Therefore, the objective function (26) minimizes the total travel time and cost, while considering the maximum possible delays set forth by the parameter Γ_k. Note that the constraints (27) are protected by (i.e., not violated by means of):

$$\beta(x_{ij}^t, \Gamma_k) = \max_{\{S^k | S^k \subseteq J^k, |S^k| \leqslant \Gamma_k\}} \sum_{(i,j) \in S^k} \hat{t}_{ij} x_{ij}^k \tag{30}$$

which is called a protection function.

Proposition 1. *Given a set of x_{ij}^{k*} values, the function (30) is equivalent to the following optimization problem:*

$$\max \sum_{(i,j) \in A} \hat{t}_{ij} x_{ij}^{k*} z_{ij}^k \tag{31}$$

$$\text{s. t.} \sum_{(i,j) \in A} z_{ij}^k \leqslant \Gamma_k \tag{32}$$

$$0 \leqslant z_{ij}^k \leqslant 1 \ \forall \ (i, j) \in A \tag{33}$$

Proof. The function (30) is to choose $\lfloor \Gamma_k \rfloor$of x_{ij}^k values at 1 and one of x_{ij}^k values at $\Gamma_k - \lfloor \Gamma_k \rfloor$. In addition, it is clear that the optimal solution of the above optimization problem (31)-(33) has $\lfloor \Gamma_k \rfloor$ variables of at 1 and one of the z_{ij}^k values is $\Gamma_k - \lfloor \Gamma_k \rfloor$, which is equivalent to the function (30) solution. □

Now, it is ready to present the following theorem.

Theorem 1. *The robust counterpart has the equivalent formulation as follows:*

$$\min \ f \sum_{k\in K}\left(\sum_{(i,j)\in A}\bar{t}_{ij}x_{ij}^{k}+\Gamma_k g_k+\sum_{(i,j)\in A^k}h_{ij}^{k}\right)+\sum_{r\in R}p_r y_r \tag{34}$$

$$\text{s. t.} \quad (2)-(10),\ (14)-(24),\ (28),\ (29)$$

$$\sum_{(i,j)\in A}\bar{t}_{ij}x_{ij}^{k}+\Gamma_k g_k+\sum_{(i,j)\in A^k}h_{ij}^{k}\leqslant t_k \forall\ k\in K \tag{35}$$

$$g_k+h_{ij}^{k}\geqslant \hat{t}_{ij}x_{ij}^{k}\ \forall\ (i,j)\in A,\ k\in K \tag{36}$$

$$h_{ij}^{k}\geqslant 0\ \forall\ (i,j)\in A,\ k\in K \tag{37}$$

$$g_k\geqslant 0\ \forall\ k\in K \tag{38}$$

where g_k and h_{ij}^{k} are dual variables.

Proof. The dual of problem (31)–(33) can be written as:

$$\min \ \Gamma_k g_k+\sum_{(i,j)\in A}h_{ij}^{k} \tag{39}$$

$$\text{s. t.} \quad g_k+h_{ij}^{k}\geqslant \hat{t}_{ij}x_{ij}^{k}\ \forall\ (i,j)\in A \tag{40}$$

$$h_{ij}^{k}\geqslant 0\ \forall\ (i,j)\in A \tag{41}$$

$$g_k\geqslant 0\ \forall\ k\in K \tag{42}$$

It is clear that the problem (31)–(33) and its dual (39)–(42) are both feasible and bounded. Then, by the strong duality, their objective values must coincide. In addition, the function (30) is equivalent to the objective function (39) by Proposition 1. Therefore by substituting (39)–(42) into problem (26)–(29), (34)–(38) are obtained. □

3.3. Heuristic methods

The ride-sharing nominal problem and its robust counterpart can be solved using off-the-shelf software if the problem size is small enough. For larger problems, decent heuristic algorithms need to be used. This is particularly important for ride-sharing problems, as the information comes in short notice and the matching between drivers and riders should be decided quickly. Therefore, the heuristic algorithm to solve the problem within a given time limit is proposed, which is shown in Algorithm 1.

Algorithm 1. Algorithm for the Ride-Sharing Problem

Algorithm for the Ride-Sharing Problem
Implement the insertion algorithm to construct a good feasible solution s^I;
Set s^I as the initial solution for the tabu search;
while *elapsed CPU time* $<$ *max CPU time* **do**
  Implement one iteration of the tabu search method;
  Update best-so-far solution s^B;
end
Return s^B

Note that the insertion algorithm is used to find the good initial solution. Indeed, it has been used for solving various vehicle routing problems (Campbell & Savelsbergh, 2004; Campbell, Vandenbussche, & Hermann, 2008), which is used for our ride-sharing problem. It constructs a reasonably good feasible solution by repeatedly and greedily inserting an unrouted rider into a partially constructed feasible solution. The constructed solution from insertion algorithm is not guaranteed to be optimal solution or near-optimal solution. Therefore, it is used as an initial solution for the tabu search method that is implemented until the terminal criterion, the maximum allowed CPU time in our case, is met.

3.3.1. Insertion algorithm

The insertion algorithm is used to find the initial feasible solution. In this paper, the insertion algorithm (Campbell & Savelsbergh, 2004) is extended to consider the capacity constraints (3), the maximum requests constraints (4), the maximum time constraints (11), and the riders that are left unserved. The details of our revised insertion algorithm is shown in Algorithm 2. A route is defined as an ordered set of nodes (locations): the route k for driver k starts with driver's origin node b_k and ends with his destination node w_k. The number of riders' od pairs between b_k and w_k is denoted by n_k, $0\leqslant n_k\leqslant m_k$, and the route k is denoted by (b_k,…, b_r,…, w_r,…, w_k). An index i is used to represent the i^{th} position (node) in route k and, as route k is for driver k, there are $|K|$ number of routes. Note that b_r and w_r should be in the same route and the position of b_r is prior to the position of w_r, $\forall\ r\in R$. The set of routes is denoted by K'. The seats that have not been taken in the vehicle of driver k is denoted by c^k. A variable E is used to keep track of the number of routes that cannot accommodate the demand d_r of rider r due to the capacity constraints (3) or the maximum requests constraints (4). The current objective function value for K' is denoted by f'. Let $K'_{i,r,k}$ denote the set of routes after rider r's origin and destination are inserted between positions $i-1$ and i in route k. The objective function value of $K'_{i,r,k}$ is denoted by $f'_{i,r,k}$ and let $\delta'=f'_{i,r,k}-f'$.

Algorithm 2 starts with $|K|$ routes, each having driver's original and destination only. Algorithm 2 stops when all nodes are either inserted into routes or added to the unserved list U' (all riders must be either picked up or put in the unserved list). The capacity constraints (3), the maximum requests constraints (4), and the maximum time constraints (11) are checked during the insertion algorithm. The time window constraints (14)–(20) are checked after the insertion algorithm ends. If the time window constraints are not satisfied, the objective function value for K' is $f'+M\lambda$where λ represents the number of routes that do not satisfy the time window constraints. Therefore, the solution K' constructed by Algorithm 2 satisfies the capacity constraints (3), the maximum requests constraints (4), and the maximum time constraints (11), but may not violate the time window constraints. As M is a very large number, $M\lambda$is considered as the penalty due to the violation of the time window constraints.

Algorithm 2. Extended Insertion Algorithm for Ride-sharing

Extended Insertion Algorithm for Ride-sharing
K' is initialized to contain $|K|$ routes, route k=(b_k, w_k), $f' = 0$, $c^k = q_k, \forall k \in K'$;
while $R \neq \emptyset$ **do**
  $\delta^\star = \infty$;
  for $r \in R$ **do**
    $E = 0$;
    for $k \in K'$ **do**
      if $c^k \geq q_r$ *and* $n_k < m_k$ **then**
        for $i \in$ *route* k **do**
          $K'_{i,r,k} = K'$, insert b_r and w_r between position $i-1$ and position i of route k for $K'_{i,r,k}$;
          $\delta' = f'_{i,r,k} - f'$;
          if $K'_{i,r,k}$ *satisfies the maximum time constraints (11)* **then**
            if $\delta' < \delta^\star$ **then**
              $\delta^\star = \delta', r^\star = r, i^\star = i, k^\star = k$
            end
          end
        end
      else
        $E = E + 1$
      end
    end
    if $E = |K|$ **then**
      $U' = U' \cup \{r\}$, $R = R \setminus r$
    end
  end
  Insert $b_{r^\star}$ and $w_{r^\star}$ between position $i^\star - 1$ and position $i^\star$ of route $k^\star$ for K';
  $f' = f' + \delta^\star$, $c^{r^\star} = c^{r^\star} - d_{r^\star}$, $R = R \setminus r^\star$;
end
Return K', f'

3.3.2. Tabu search

In tabu search, an initial solution and a tabu list length are to be determined. In this paper, the initial solution can be found by implementing the extended insertion algorithm. Four move operators are used to search neighbor solutions. Subsequently, all the neighbor solutions are evaluated and ranked from smallest objective function value to largest objective function value.

The four move operators are: (1) Exchange-within-route: Choose location i and location i' in route k and switch the their positions. This move operator is used $\forall\, i, i' \in$route k, $i \neq i'$, $\forall\, k \in K'$; (2) Exchange-between-routes: If b_r and w_r are in route k and b'_r and w'_r are in route k', switch the positions of b_r and b'_r, and switch the positions of w_r and w'_r, $\forall\, r, r' \in R$, $\forall\, k, k' \in K'$; (3) Relocate-between-routes: Remove b_r and w_r in route k, insert b_r between position $i-1$ and position i in route k', insert w_r between position $i'-1$ and position i' in route k'. This move operator is applied when $i < i', \forall\, i \leqslant n_k + 1, \forall\, i \leqslant n_k + 2, \forall\, r \in R, \forall\, k \in K'$; and (4) Exchange-unserved-customer: If b_r and w_r are in route k and r' is in the unserved list U', replace b_r with b'_r and replace w_r with w'_r, remove r' from U' and add r to U'. This move operator is applied $\forall\, r \in R$, $\forall\, k \in K'$, $\forall\, r' \in U'$.

After a neighbor solution is found, the capacity constraints (3), the maximum requests constraints (4), the maximum time constraints (11) are checked. If the solution does not satisfy these constraints, the solution is not considered. If the solution does not satisfy time window constraints, the objective function value for K' is set to be $f' + M\lambda$. The tabu search algorithm is shown in Algorithm 3.

In Algorithm 3, the current solution is denoted by $K'$and the tabu list is denoted by σ. The best-so-far solution during the implementation process is denoted by K^{best}, and the objective function value of K^{best} is denoted by f^{best}. In addition, the neighbor solution of K' is denoted by K'_h, and f'_h is the objective function value of K'_h. The set of K'_h is denoted by M_k, and K'_h are rearranged from the one with the smallest f'_h to the one with largest f'_h. The maximum CPU time that allows program to run is denoted by $B_{max}$and the elapsed CPU time is denoted by B. While $B < B_{max}$, the tabu search is implemented iteratively.

Algorithm 3. Tabu Search Algorithm

Tabu Search Algorithm
K' = current solution, $K^{best} = K'$, $\sigma = \emptyset$;
while $B < B_{max}$ **do**
  $M_k = \emptyset$;
  Find all K'_h of K' according to the move operators;
  Add the K'_h that satisfy constraints (3), (4), and (11) into M_k;
  for $K'_h \in M_k$ **do**
    Evaluate f'_h of K'_h;
  end
  Rank $K'_h \in M_k$ from the smallest f'_h to the largest f'_h;
  for $K'_h \in M_k$ **do**
    if $K'_h \notin \sigma$ **then**
      $K' = K'_h$;
      Add K'_h to σ;
      if $f'_h < f^{best}$ **then**
        $K^{best} = K'_h$;
        $f^{best} = f'_h$;
      end
    else
      if $f'_h < f^{best}$ **then**
        $K' = K'_h$;
        add K'_h to σ;
        $K^{best} = K'_h$;
        $f^{best} = f'_h$;
      end
    end
    Break the for-loop when K' is updated;
  end
  Update σ based on frequency
end
Return K^{best} and f^{best}

3.3.3. Algorithm for solving robust counterparts

To solve the robust counterpart of the ride-sharing model to consider uncertain travel times, Algorithm 4 is proposed in which the total amount of increased travel time due to the parameter Γ_k, $\forall\, k \in K$ is considered. This algorithm can be used in conjunction with the insertion algorithm and the tabu search. That is, for the current solution K' in tabu search, this Algorithm 4 can be applied for calculating the increased travel time due to uncertainty for the robust counterpart. Depending on the degree of robustness (that can be decided by values of Γ_k, $\forall\, k \in K$), $\hat{t}_{ij}$ for some (i, j) in A, may or may not be included in the solution. Algorithm 4 explicitly takes into account such cases.

Algorithm 4. Compute Total Increased Travel Time Based on Γ_k

Compute Total Increased Travel Time Based on Γ_k
Given $\Gamma_k, \forall k \in K$, $\hat{t}_{ij}, (i,j) \in A$, $\hat{t}_{ij}, (i,j) \in A$, and a solution K';
for $k \in K$ **do**
  Sort all $\hat{t}_{ij}$ in route k in decreasing order;
  l_k = the position index of $\hat{t}_{ij}$ in sorted order;
  S^k = the set of arcs of which the $\hat{t}_{ij}$ are considered in route k based on Γ_k. $S^k = \emptyset$;
  for $l_k \leq \Gamma_k$ **do**
    if $\hat{t}_{ij} + \hat{t}_{ij}$ *does not violate any constraint* **then**
      Add arc (i,j) of $\hat{t}_{ij}$ at position l_k into S^k;
    end
    if *If no more arc left* **then**
      break;
    end
  end
  Total increased travel time in route $k = \sum_{(i,j)\in S^k} \hat{t}_{ij}$;
end
Total increased travel time in $K' = \sum_{k\in K} \sum_{(i,j)\in S^k} \hat{t}_{ij}$

4. Cluster-first-route-second approach

For a ride-sharing problem, it is important to achieve computational tractability, as quick matching and routing decisions are the key to the success. Besides using the heuristic algorithms presented in the previous section to solve the ride-sharing problem, a cluster-first-route-second approach is proposed. For a large scale problem consisting of hundreds or thousands of drivers and riders, it will still be computationally very challenging even if the heuristic methods presented above are used. To resolve such a computational issue, the problem is decomposed into multiple small problems by clustering approaches. In our problem setting, the capacity of personal vehicle is in general four to five, which

implies that each driver's problem will consist of a small number of rider nodes such that it can achieve computational tractability. The size of each cluster will be decided depending on the clustering method used. In this paper, two clustering approaches are considered: greedy algorithm and *k*-means approach, which are illustrated in Sections 4.1 and 4.2. After the clustering is done, each sub-group will contain one driver and several riders, which then can be solved either by the heuristic methods (insertion with tabu search) or off-the-shelf software. The off-the-shelf software is used after clustering as the number of rider nodes is small enough in the ride-sharing context.

4.1. Greedy algorithm

The greedy algorithm proposed in this section considers the balance of assigning riders to drivers. After the driver k is assigned to a rider, then driver $k+1$ will be assigned to another rider. That means the drivers take turn to be assigned to riders. The list of rider waiting for the assignment is denoted by U. The greedy algorithm is shown in Algorithm 5.

Algorithm 5. Greedy Algorithm

Greedy Algorithm — Initialization: $U = R$, $k = 1$, $numRequest_k = 0$
while $U \neq \emptyset$ **do**
  Find the rider r in U with nearest origin and destination to driver k;
  if $numRequest_k \leq m_k$ **then**
    Assign rider r to driver k if m_k;
    $numRequest_k = numRequest_k + 1$;
    Remove rider r from U;
  end
  if $k = |K|$ **then**
    $k = 1$;
  else
    $k = k + 1$;
  end
end

4.2. k-means algorithm

To compare with the greedy algorithm, *k*-means algorithm is employed as an alternative to form clusters of riders. When the *k*-means algorithm is used, the riders are divided into $|K|$ clusters without considering drivers. Then, each cluster is assigned to one driver based on the distances between each cluster and drivers. The details of *k*-means algorithm is shown in Algorithm 6. The driver-cluster assignment algorithm needs to be implemented after the clustering is done, which is shown in Algorithm 7.

Algorithm 6. *k*-means Algorithm

***k*-means Algorithm** — Initialization: q' (index set of current centroids) $\leftarrow$ randomly chosen $|K|$ riders;
q'_n (index set of new centroids) $\leftarrow \emptyset$;
$stop = False$;
while $stop = False$ **do**
  For each rider r, find the nearest current centroid from q';
  Assign rider r to to the nearest centroid;
  After assigning all riders, calculate the new centroid of each cluster;
  $q'_n \leftarrow$ new centroids;
  if $q'_n = q'$ **then**
    $stop = True$;
  else
    $q' \leftarrow q'_n$;
  end
end
Return q';

Algorithm 7. Driver-Cluster Assigning Algorithm

Driver-Cluster Assigning Algorithm — Initialization: Given $|K|$ drivers, let $k = 1$;
Let UC be the set of unassigned clusters;
while $k \neq |K|$ **do**
  For driver k, find the nearest centroid i in UC;
  Assign the cluster i to driver k;
  $UC \leftarrow UC \setminus i$;
  $k = k + 1$;
end

5. Numerical examples

In this section, the proposed algorithms are implemented and analyses for the outcomes are provided. Four benchmark instances from Augerat et al. (1995) are used. These benchmark problems are named as P-n16-k8 (16 nodes), A-n32-k5 (32 nodes), A-n44-k6 (44 nodes), and A-n101-k10 (101 nodes). For each benchmark instance, the coordinates of the nodes are used to calculate the travel distance between each pair of nodes (in the unit of kilometers), then let the nominal travel time, $\bar{t}_{ij}$, be the same as travel distance by assuming the average vehicle speed is fixed as 60 km/hour (Then, it takes 1 min to travel 1 km.) for the sake of simplicity.

Algorithm 8. Nearest Neighbor Algorithm

Nearest Neighbor Algorithm —
Initialization:
For each driver k, let $numRequest_k = 0, P_k = \{b_k, w_k\}$;
$k = 1$;
while $numRequest_k \leq m_k, \sum_{r \in R} d_r a_{kr} \leq q_k$ **do**
  For driver k;
  Starting from b_k, find the nearest rider r in R;
  Assign the rider r to driver k. That is, $P_k \leftarrow P_k \cup \{b_r, w_r\}$;
  b_r will be the next starting node;
  $R \leftarrow R \setminus r$;
  if $k = |K|$ **then**
    k=1;
  else
    $k = k + 1$;
  end
end
for $k \in K$ **do**
  Given that the origin is b_k and the destination is w_k;
  Find a shortest path using the nodes in P_k;
end

Each benchmark problem (nominal and its robust counterpart) is solved four times: (1) No clustering, exact algorithm, (2) No clustering, heuristic methods (insertion plus tabu search), (3) Cluster-first-route-second, greedy algorithm, and (4) Cluster-first-route-second, k-means algorithm, except the two problems: 16-node and 101-node problem. In the 16-node problem, a simple nearest neighbor algorithm shown in Algorithm 8 is used in addition to the four methods for comparison purposes. The 101 node problem is too large for the exact solution and, therefore, only the three remaining heuristic methods are used. The off-the-shelf solver, Gurobi, is used to solve the problems to find an exact solution and Python is used for all heuristic methods. Let us present some key results in this section. Some results are omitted here due to space limitation but the detailed results for all cases are summarized in Tables 6–17 in Appendix.

For the 16 node case, two scenarios are considered: the one with different origins and destinations and the other with different origins but the same destination. In the first scenario, $|K| = 3$, $|R| = 5$. The capacity of each vehicle (excluding driver seat) is 4, and the origins and destinations of drivers and riders are all different. The node index is randomly assigned to b_k, w_k, b_r, and w_r as shown in Table 2. In this scenario, $p_r = 100$, $e_{b_r} = 0$, $l_{b_r} = 50$, $e_{w_r} = 0$, $l_{w_r} = 100$, $\forall\, r \in R$. For drivers, $q_k = 4$, $m_k = 4$, $T_k = 110$, $\forall\, k \in K$. The coefficient f to convert travel time into travel cost is assumed to be 1. In this scenario, the outcomes of the problem are shown in Fig. 1 and Table 3. For this example, it is better to solve the model using Gurobi because an optimal solution can be found, as the size of the problem is small (16 nodes). However, other heuristic method results are shown for the comparison purposes. As shown in Table 2, heuristic (insertion + tabu), k means clustering, and greedy clustring results are worse than the exact method results by 7%, 14%, and 29%, respectively, with respect to the objective function value. The heuristic methods (including clustering approach) need to be avoided for small size problem.

In scenario 2, drivers and riders depart from different origins but want to arrive at the same destination. This scenario can be found in several real-world problems. For example, participants may want to go to the same bus terminal from different origins. In another example, participants who work at the same company can arrange ride-sharing daily. For disaster management, ride-sharing for evacuation from dangerous zones to a shelter may be arranged. In this scenario, the demand of each rider request is set to be 1 (one rider per request), and two cases are considered: $|K| = 2$ and $|K| = 3$, as shown in Figs. 2 and 3. Node 16 is the destination in the case. For $|K| = 2$, node 1 and node 2 are the origins of drivers and other nodes are the origins of riders. For $|K| = 3$, nodes 1, 2, and 3 are the origins of drivers and other nodes are the origins of riders. The results are summarized in Table 4 and Figs. 2 and 3. As expected, the exact method provides the best solution for both cases: $|K| = 2$ and $|K| = 3$, and the insertion plus tabu heuristic results are excellent whose objective function values are at most 3.7% worse than the optimal solution. Greedy and k-means methods perform very good for $|K| = 2$ but not for $|K| = 3$. The nearest neighbor results can be considered as the worst because it is based on myopic search method, as shown in Figs. 2,3. As it is clear that the nearest neighbor method is much worse than other ones, it will be excluded for the remaining

Table 2
Settings for 16 node case, first scenario.

	Rider r					Driver k		
	1	2	3	4	5	1	2	3
Node i for origin	7	8	9	10	11	1	2	3
Node i for destination	12	13	14	15	16	4	5	6
d_r	2	1	2	1	1			

Table 3
Results for 16 node case, first scenario.

Method	Objective function value	Increased value	%	CPU time (second)
Exact	150.35	–	–	0.03
Heuristic	160.46	10.11	7%	0.01
Greedy algorithm	194.20	43.86	14%	0.82
k-means algorithm	170.68	20.33	29%	0.69

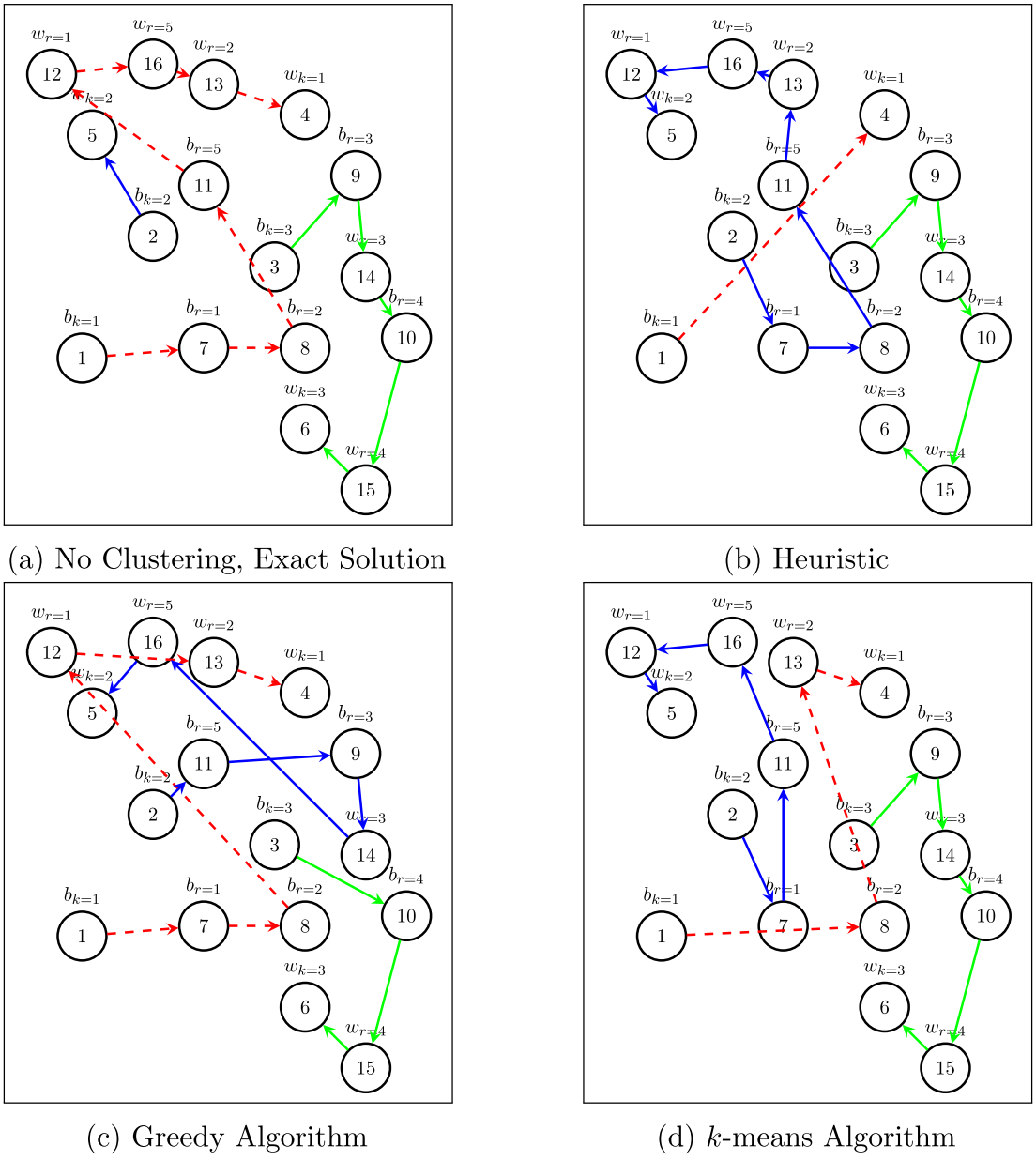


Fig. 1. Different origins and destinations, 16 node benchmark problem, $|K|=3$.

examples. Note that the k-means result is the worst in $|K| = 3$ case because of the unserved riders, which can be much better for larger problems as there will be more neighboring riders in each cluster (thereby, less unserved riders).

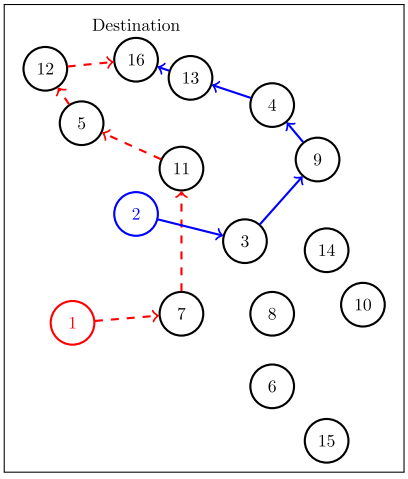
For the robust model, the maximum delay is calculated as follows.

$$\hat{t}_{ij} = 0.1\bar{t}_{ij} + 5 i \in V, j = 1, ...,8 \quad (43)$$

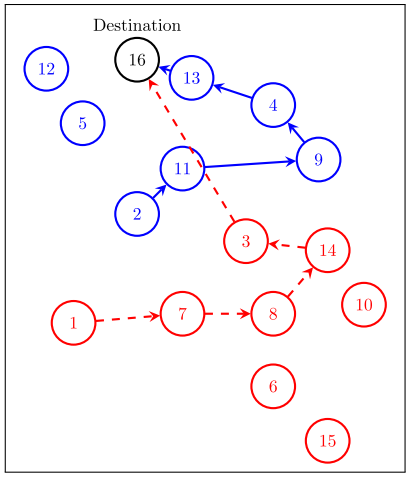
$$\hat{t}_{ij} = 0.2\bar{t}_{ij} i \in V, j = 9, ...,16 \quad (44)$$

The 16 node case outcomes are illustrated in Figs. 2,3 and detailed results including objective function values and route information are shown in Tables 6–8 in the Appendix. The route graphs for 32 node and 44 node problems are omitted here. Fig. 2 shows the $|K| = 2$ case in which nodes 1 and 2 are the driver nodes, nodes 3 to 15 are rider nodes, and node 16 is the destination. Fig. 3 shows the $|K| = 3$ case in which nodes 1 to 3 are the driver nodes, nodes 4 to 15 are rider nodes, and nodes 16 is the destination. When $|K| = 2$, the exact solution is the same as the heuristic solution for the non-clustering case, and the greedy solution is same as k-means solution for the clustering case, which is due to its small size. When $|K| = 3$, all four solutions are different. Note in particular that there are two riders not picked up in the k-means solution when $|K| = 3$, while all riders are picked up in other solutions, which has a great impact on the objective function value that is shown in Fig. 4.

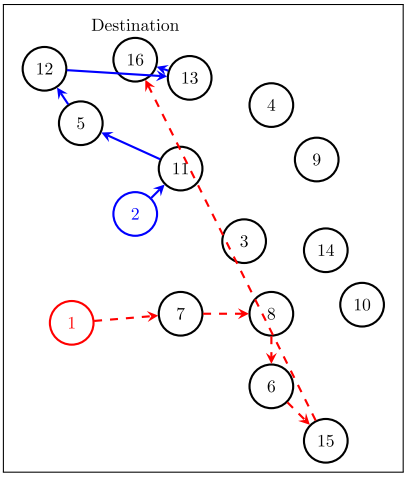
For the nominal problem, the changes in objective function value for heuristic solutions compared to the base model (non-clustering exact solution) are illustrated in Fig. 4 for 16, 32, and 44 node cases. For the 101 node problem, only one case $|K| = 10$ is considered, whose results are shown in Table 5, because small $|K|$ cases are meaningless (most riders will be left unserved.). In each graph in Fig. 4, the leftmost group represents non-clustering heuristic case (insertion algorithm with tabu search, maximum allowable CPU time: 1 h). The greedy and k-means solutions are shown in the second and third group, respectively, from the left. It is clear that the non-clustering exact solution is always the best but it takes more than 2 days to solve the 44 node problem with the Gurobi solver (Windows 10, 12 GB RAM, 4th Generation Intel Quad Core i7-4712HQ), which makes it impractical for our ride-sharing problem. In contrast to the exact method, it takes no more than 1 min



(a) No Clustering, Exact Solution and Heuristic



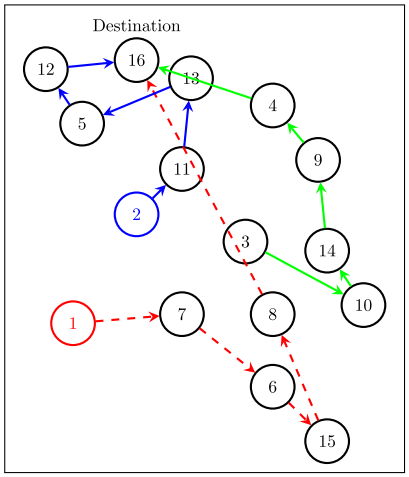
(b) Greedy Algorithm and k-means Algorithm



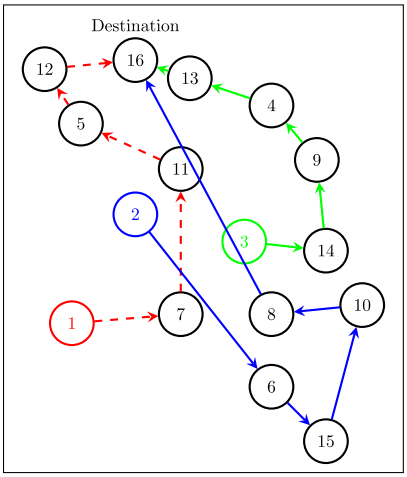
(c) Nearest Neighbor Algorithm

Fig. 2. 16 node benchmark problem, $|K|=2$.

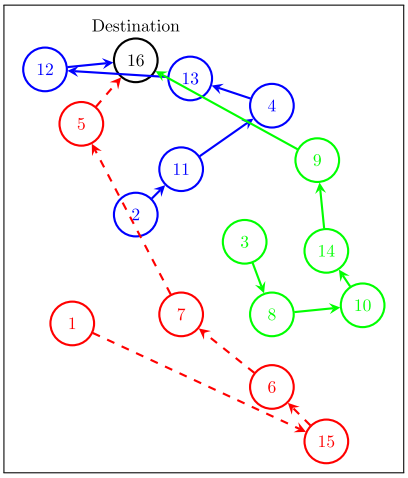
(CPU time) when the greedy and k-means algorithms are implemented even for the largest example with 101 node robust counterpart. Note also that the non-clustering heuristic algorithm (insertion algorithm with tabu search) results are in general better than clustering solutions but maximum 1 h CPU time is allowed when the algorithm is implemented. The insertion algorithm in conjunction with the tabu search may not produce meaningful results if the maximum CPU time allowed is 2 s.



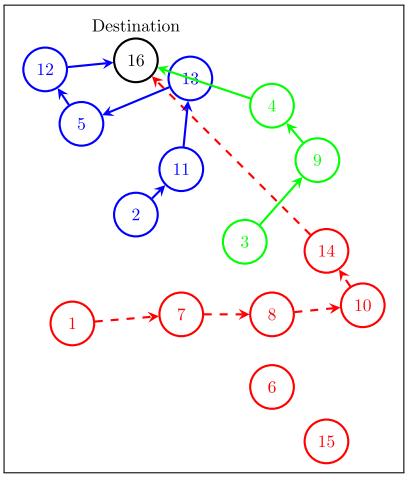
(a) No Clustering, Exact Solution



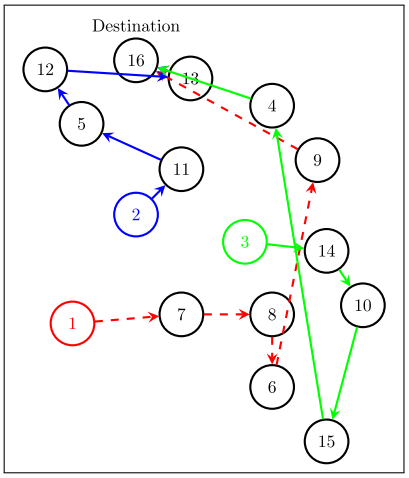
(b) No Clustering, Heuristic Solution



(c) Greedy Algorithm



(d) k-means Algorithm



(e) Nearest Neighbor Algorithm

Fig. 3. 16 node benchmark problem, $|K|=3$.

Table 4
Results for 16 node case, second scenario.

	$\|K\| = 2$		$\|K\| = 3$	
Method	Obj. func. value	Increased %	Obj. func. value	Increased %
Exact	605.4	–	183.4	–
Heuristic	605.4	0.0%	190.1	3.7%
Greedy algorithm	609.4	0.7%	282.3	53.9%
k-means algorithm	609.4	0.7%	352.0	91.9%
Nearest neighbor	714.1	18.0%	331.4	80.7%

Fig. 5 shows how the objective value changes in exact solutions as the degree of uncertainty, Γ_k, increases in the robust counterpart. Here Γ_k represents, for each driver k, the number of arcs subject to uncertainty. Therefore, if $\Gamma_1 = 3$, it implies that driver 1 will face up to 3 arcs that are subject to uncertainty during his/her course from the origin to the destination. If $\Gamma_k = 0, \forall\ k$, then the problem will become deterministic (same as the nominal case). If greater Γ_k is allowed, then there will be more arcs with a delay. Consequently, the objective function value will increase. However, as it is clear from the results, the increase in the objective value is minimal (up to 2.61%) for 32 node and 44 node cases, and it is even smaller (up to 1.34%) in the 101 node case. This is because the number of potential arcs to be visited by a single

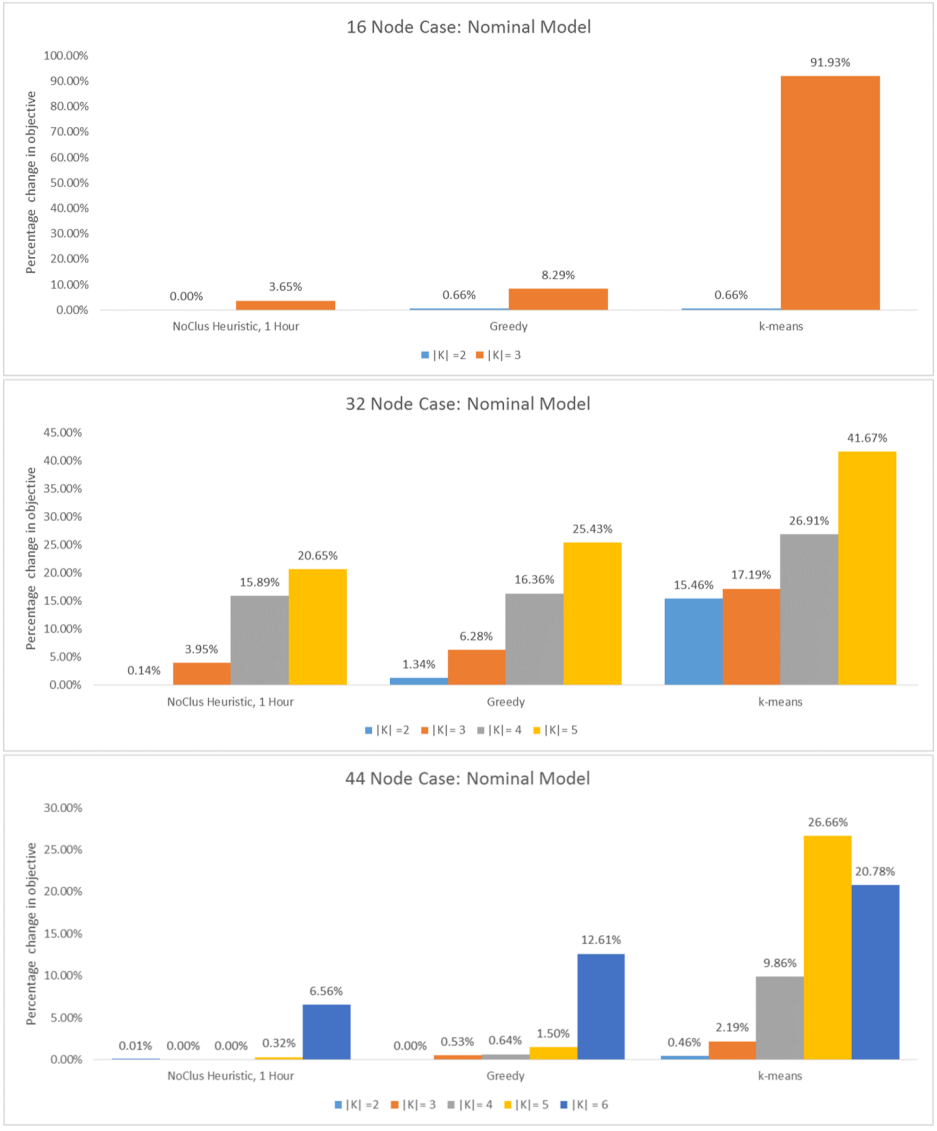


Fig. 4. Percentage change in objective, base: no clustering exact solution.

Table 5
Results for 101 node case, $|K| = 10$.

Method	Objective function value	Increased value	%	CPU time
Heuristic	5390.1	–	–	1 h
Greedy algorithm	5482.3	92.2	1.7%	Less than 1 min
k-means algorithm	6079.7	689.6	12.8%	Less than 1 min

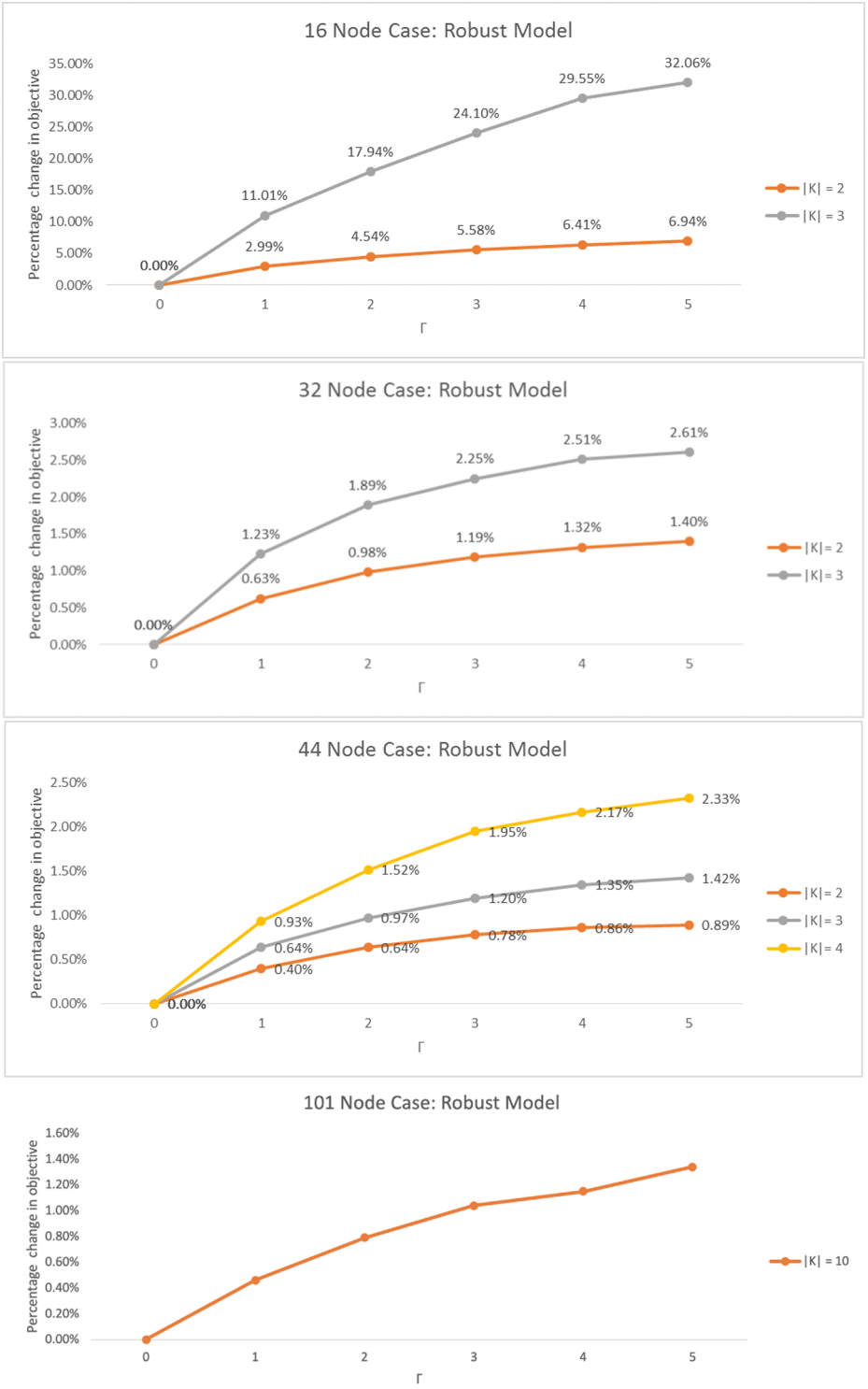


Fig. 5. Percentage Change in Objective, Base: Nominal Solution.

driver (that is, the number of all rider nodes divided by number of drivers. For example, it is 11(=44/4) for 44 node, $|K|=4$ case) is much greater than the vehicle capacity q_k that is 4 in our case study. In this case, out of all the arcs connecting the driver and rider nodes, up to Γ_k are subject to uncertainty and there still remain a substantial number of arcs not subject to uncertainty (not experiencing a delay). Therefore, the driver will choose those arcs as many as possible, trying to minimize the increase in the objective function value. However, it goes up to 32% for the 16 node case when $|K| = 3$. This is because most of the arcs each driver can potentially visit are all subject to uncertainty (experiencing delays), and, therefore, a driver cannot but visit such arcs with delays.

6. Conclusion and future work

In this paper, a new model for ride-sharing is proposed to address different origins and destinations of drivers and riders, which can determine the match between drivers and riders and their routes quickly. In addition, a robust counterpart is constructed to address the travel time uncertainty. To solve large scale problems, an extended insertion algorithm and tabu search-based heuristic is developed. As one of the

keys to the success of the ride-sharing is an instant match upon request, a cluster-first-route-second approach is proposed, which is compared with the heuristic algorithm (insertion + tabu) in terms of the computational tractability and solution quality. The clustering methods, the greedy and *k*-means algorithms, can find reasonably good solutions quickly even for the largest 101 node nominal problem and its robust counterpart.

Exact solutions found by solvers are optimal but it takes several hours to several days for 32 node and 44 node benchmark problems and it was not possible for the 101 node problem. This is clearly not acceptable for the ride-sharing problem. The insertion method with tabu search solutions are slightly worse than the optimal solutions but in general slightly better than other heuristic (greedy algorithm and *k*-means) solutions implemented with clustering methods.

Regarding the solutions from cluster-first-route-second approach, the results show that the greedy method outperforms *k*-means method for most instances of benchmark problems. The results also show that if a single cluster size is large enough (10 or more nodes in one cluster) than the vehicle capacity (5 including the driver), the objective deterioration compared with the optimal solutions is fairly minor (up to 6.3%) in our benchmark problems. This is mainly because there are more choices available for each driver to visit riders (choosing the riders up to its capacity out of a lot more riders within its cluster) such that the driver can choose cost minimizing riders. The same idea can be applied to the robust counterpart. The driver will choose arcs that are less likely subject to delays. Therefore, the clustering methods and robust optimization approach can be particularly useful when the number of drivers are smaller than the number of riders to be served.

For future work, more sophisticated clustering approaches can be considered to improve the quality of solutions. In addition, a new exact approach employing the state-of-the-art such as modern column-generation, relaxation, and/or decomposition approaches can be proposed. Furthermore, it will be a natural extension to include riders' travel time/cost in the objective function. Relaxation of the assumption that a rider is not allowed to get off before all the pick-ups are done will be an interesting extension as well. Last, an en-route ride-sharing problem can be of particular interest. In the en-route ride-sharing problem, a drive may or may not accept riders' request depending on the driver's acceptance criteria, for which stochastic dynamic programming can be a useful tool to handle the problem.

CRediT authorship contribution statement

Yinglei Li: Software, Writing - original draft, Validation, Visualization, Data curation. **Sung Hoon Chung:** Conceptualization, Methodology, Writing - review & editing, Supervision, Investigation.

Appendix A. Detailed results

Tables 6– 17.

Table 6
Results of 16 node case: nominal model.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\|K\| = 2$	Obj value	605.4	605.4	609.4	609.4
	Route 1	1,7,11,5,12,16	1,7,11,5,12,16	1,7,8,14,3,16	1,7,8,14,3,16
	Route 2	2,3,9,4,13,16	2,3,9,4,13,16	2,11,9,4,13,16	2,11,9,4,13,16
$\|K\| = 3$	Obj value	183.4	190.1	198.6	352.0
	Route 1	1,7,6,15,8,16	1,7,11,5,12,16	1,15,6,7,5,16	1,7,8,10,14,16
	Route 2	2,11,13,5,12,16	2,6,15,10,8,16	2,11,4,13,12,16	2,11,13,5,12,16
	Route 3	3,10,14,9,4,16	3,14,9,4,13,16	3,8,10,14,9,16	3,9,4,16

Table 7
Results of 16 node case: robust model, $|K| = 2$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	605.4	605.4	609.4	609.4
	Route 1	1,7,11,5,12,16	1,7,11,5,12,16	1,7,8,14,3,16	1,7,8,14,3,16
	Route 2	2,3,9,4,13,16	2,3,9,4,13,16	2,11,9,4,13,16	2,11,9,4,13,16
$\Gamma_k = 1$	Obj value	623.5	630.6	625.1	625.1
	Route 1	1,7,8,3;4,16	1,7,3,5;12,16	1;7,8,14,3,16	1;7,8,14,3,16
	Route 2	2,11,13,12;5,16	2,11,9,4,13;16	2,11,13,12;5,16	2,11,13,12;5,16
$\Gamma_k = 2$	Obj value	632.9	643.2	634.4	634.4
	Route 1	1;7,8,3;4,16	1;7;3,12,5,16	1;7;8,14,3,16	1;7;8,14,3,16
	Route 2	2,11,13;12;5,16	2,11,9; 4,13;16	2,11,13;12;5,16	2,11,13;12;5,16
$\Gamma_k = 3$	Obj value	639.2	652.0	642.3	642.3
	Route 1	1;7;14,9;4,16	1;7;3;12,5,16	1;7;8,14;3,16	1;7;8,14;3,16
	Route 2	2,11;13;12;5,16	2,11;9;4,13;16	2,11;13;12;5,16	2,11,13,12,5,16
$\Gamma_k = 4$	Obj value	644.2	653.6	648.8	648.8
	Route 1	1;3,14;9;4;16	1;7;3;11;5,16	1;7;8,14;3;16	1;7;8,14;3;16
	Route 2	2,11;13;12;5;16	2;14;9;4,13;16	2,11;13;12;5;16	2,11;13;12;5;16
$\Gamma_k = 5$	Obj value	647.4	657.3	652.0	652.0
	Route 1	1;3;14;9;4;16	1;7;3;11;5;16	1;7;8;14;3;16	1;7;8;14;3;16
	Route 2	2;11;13;12;5;16	2;14;9;4;13;16	2;11;13;12;5;16	2;11;13;12;5;16

Note: a semi-colon is used for arcs that are subject to uncertainty (experiencing a delay) while a comma is used for arcs not subject to uncertainty.

Table 8
Results of 16 node case: robust model, $|K| = 3$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	183.4	190.1	282.3	352.0
	Route 1	1,7,6,15,8,16	1,7,11,5,12,16	1,6,7,5,16	1,7,8,10,14,16
	Route 2	2,11,13,5,12,16	2,6,15,10,8,16	2,11,4,13,12,16	2,11,13,5,12,16
	Route 3	3,10,14,9,4,16	3,14,9,4,13,16	3,8,10,14,9,16	3,9,4,16
$\Gamma_k = 1$	Obj value	203.6	221.6	301.8	378.6
	Route 1	1,7,6,15;8,16	1;7,11,5,12,16	1,6,7;5,16	1;7,8,10,14,16
	Route 2	2,11,13,12;5,16	2,10,15,6,8; 16	2,11;4,13,12,16	2,11,13,12;5,16
	Route 3	3,10,14,9;4,16	3,14,9,4,13;16	3;8,10,14,9,16	3;9,4,16
$\Gamma_k = 2$	Obj value	216.3	244.7	316.9	393.6
	Route 1	1,7,6,15;8;16	1,7;11,5;12,16	1;6,7;5,16	1;7;8,10,14,16
	Route 2	2,11,13;12;5,16	2,10,15;6,8;16	2,11;4,13;12,16	2,11,13;12;5,16
	Route 3	3,10,14,9;4;16	3,14,9;4,13;16	3;8,10,14,9;16	3;9;4,16
$\Gamma_k = 3$	Obj value	227.6	258.1	327.2	404.7
	Route 1	1,7;6,15;8;16	1;7,8;5;12,16	1;6;7;5,16	1;7;8,10,14;16
	Route 2	2,11;13;12;5,16	2;6,15;10;11,16	2,11;4,13;12;16	2,11;13;12;5,16
	Route 3	3;10,14,9;4;16	3,14;9;4,13;16	3;8;10,14,9;16	3;9;4;16
$\Gamma_k = 4$	Obj value	237.6	268.6	332.9	408.5
	Route 1	1;7;6,15;8;16	1;7;8;12;5,16	1;6;7;5;16	1;7;8;10,14;16
	Route 2	2,11;13;12;5;16	2;6,15;10;11;16	2,11;4;13;12;16	2,11;13;12;5;16
	Route 3	3;10,14;9;4;16	3,14;9;4;13;16	3;8;10,14;9;16	3;9;4;16
$\Gamma_k = 5$	Obj value	242.2	273.9	335.8	411.4
	Route 1	1;7;6;15;8;16	1;7;8;12;5;16	1;6;7;5;16	1;7;8;10;14;16
	Route 2	2;11;13;12;5;16	2;6;15;10;11;16	2;11;4;13;12;16	2;11;13;12;5;16
	Route 3	3;10;14;9;4;16	3;14;9;4;13;16	3;8;10;14;9;16	3;9;4;16

Table 9
Results of 32 node case: nominal model.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\|K\| = 2$	Obj value	2238.3	2241.5	2268.2	2584.4
	Route 1	1,31,27,17,22,32	1,31,17,13,22,32	1,28,25,15,27,32	1,31,17,8,22,32
	Route 2	2,8,14,18,20,32	2,8,14,18,20,32	2,8,14,18,20,32	2,4,32
$\|K\| = 3$	Obj value	1836.7	1909.2	1952.0	2152.5
	Route 1	1,31,27,17,13,32	1,31,27,17,13,32	1,28,25,15,27,32	1,31,27,17,13,32
	Route 2	2,8,14,18,20,32	2,8,14,22,32	2,8,14,18,20,32	2,8,14,18,20,32
	Route 3	3,4,24,7,22,32	3,24,4,18,20,32	3,4,24,7,32	3,29,32
$\|K\| = 4$	Obj value	1573.6	1823.6	1831.1	1997.1
	Route 1	1,28,25,15,8,32	1,31,27,8,14,32	1,28,25,15,27,32	1,31,27,17,8,32
	Route 2	2,13,31,27,17,32	2,13,17,22,32	2,13,17,8,22,32	2,14,22,18,20,32
	Route 3	3,24,29,18,20,32	3,7,20,32	3,24,14,32	3,24,29,32
	Route 4	4,7,14,22,32	4,24,29,18,32	4,7,18,32	4,32
$\|K\| = 5$	Obj value	1383.6	1669.3	1735.4	1960.1
	Route 1	1,28,25,15,8,32	1,31,27,8,14,32	1,28,25,27,32	1,31,27,17,8,32
	Route 2	2,13,31,27,17,32	2,13,17,22,32	2,13,17,8,22,32	2,14,22,18,20,32
	Route 3	3,24,29,18,20,32	3,18,32	3,24,14,20,32	3,24,7,32
	Route 4	4,7,14,22,32	4,7,20,32	4,7,18,32	4,32
	Route 5	5,9,19,32	5,12,29,24,32	5,12,32	5,32

Table 10
Results of 32 node case: robust model, $|K| = 2$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	2238.3	2241.5	2435.2	2615.1
	Route 1	1,31,27,17,22,32	1,31,17,13,22,32	1,31,27,32	1,31,17,8,22,32
	Route 2	2,8,14,18,20,32	2,8,14,18,20,32	2,8,14,18,20,32	2,32
$\Gamma_k = 1$	Obj value	2252.3	2255.4	2452.1	2629.2
	Route 1	1,31,27,17;22,32	1,31,17,13; 22,32	1,31,27;32	1,31,17;8,22,32
	Route 2	2;8,14,18,20,32	2;8,14,18,20,32	2;8,14,18,20,32	2;32
$\Gamma_k = 2$	Obj value	2260.3	2263.4	2460.1	2634.6
	Route 1	1;31,27,17;22,32	1;31,17,13;22,32	1;31,27;32	1,31,17;14;22,32
	Route 2	2;8,14;18,20,32	2;8,14;18,20,32	2;8,14;18,20,32	2;32
$\Gamma_k = 3$	Obj value	2264.9	2268.3	2464.4	2637.8
	Route 1	1;31,27,17;22;32	1;31,17;13;22,32	1;31;27;32	1;31,17;14;22,32
	Route 2	2;8;14;18,20,32	2;8;14;18,20,32	2;8;14;18,20,32	2;32
$\Gamma_k = 4$	Obj value	2267.8	2271.2	2465.4	2639.7
	Route 1	1;31,27;17;22;32	1;31;17;13;22,32	1;31;27;32	1;31;17;14;22,32
	Route 2	2;8;14;18,20;32	2;8;14;18,20;32	2;8;14;18,20;32	2;32
$\Gamma_k = 5$	Obj value	2269.6	2273.5	2465.9	2641.5
	Route 1	1;31;27;17;22;32	1;31;17;13;22;32	1;31;27;32	1;31;17;14;22;32
	Route 2	2;8;14;18;20;32	2;8;14;18;20;32	2;8;14;18;20;32	2;32

Table 11
Results of 32 node case: robust model, $|K| = 3$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	1909.2	1909.2	2119.0	2196.0
	Route 1	1,31,27,17,13,32	1,31,27,17,13,32	1,31,27,32	1,31,27,17,13,32
	Route 2	2,8,14,22,32	2,8,14,22,32	2,8,14,18,20,32	2,8,14,18,20,32
	Route 3	3,24,4,18,20,32	3,24,4,18,20,32	3,4,24,7,32	3,32
$\Gamma_k = 1$	Obj value	1932.7	1932.7	2145.3	2221.3
	Route 1	1,31,27;8,14,32	1,31,27;8,14,32	1,31,27;32	1,31,27,17,13;32
	Route 2	2,13,17;22,32	2,13,17;22,32	2;8,14,18,20,32	2;8,14,18,20,32
	Route 3	3,24,4;18,20,32	3,24,4;18,20,32	3,4,24,7;32	3;32
$\Gamma_k = 2$	Obj value	1945.3	1945.3	2161.0	2229.4
	Route 1	1,31,27;8,14;32	1,31,27;8,14;32	1;31,27;32	1;31,27,17,13;32
	Route 2	2,13;17;22,32	2,13;17;22,32	2;8,14;18,20,32	2;8,14;18,20,32
	Route 3	3,24;4;18,20,32	3,24;4;18,20,32	3,4,24;7;32	3;32
$\Gamma_k = 3$	Obj value	1952.2	1952.2	2170.6	2234.2
	Route 1	1;31,27;8,14;32	1;31,27;8,14;32	1;31;27;32	1;31,27,17;13;32
	Route 2	2,13;17;22;32	2,13;17;22;32	2;8;14;18,20,32	2;8;14;18,20,32
	Route 3	3;24;4;18,20,32	3;24;4;18,20,32	3;4,24;7;32	3;32
$\Gamma_k = 4$	Obj value	1957.2	1957.7	2173.0	2237.0
	Route 1	1;31,27;17;13;32	1;31,27;8;14;32	1;31;27;32	1;31,27;17;13;32
	Route 2	2;8;14;22;32	2;13;17;22;32	2;8;14;18,20;32	2;8;14;18,20;32
	Route 3	3;24;4;18,20;32	3;24;4;18,20;32	3;4;24;7;32	3;32
$\Gamma_k = 5$	Obj value	1959.0	1959.6	2173.5	2238.9
	Route 1	1;31;27;17;13;32	1;31;27;8;14;32	1;31;27;32	1;31;27;17;13;32
	Route 2	2;8;14;22;32	2;13;17; 22;32	2;8;14;18,20;32	2;8;14;18;20
	Route 3	3;24;4;18;20;32	3;24;4;18;20;32	3;4;24;7;32	3;32

Table 12
Results of 44 node case: nominal model.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\|K\| = 2$	Obj value	3438.8	3439.3	3438.8	3454.7
	Route 1	1,3,23,37,10,44	1,7,4,26,30,44	1,3,23,37,10,44	1,3,23,37,10,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	2,36,21,31,24,44
$\|K\| = 3$	Obj value	2995.4	2995.4	3011.2	3061.0
	Route 1	1,7,4,26,30,44	1,7,4,26,30,44	1,32,5,35,18,44	1,7,4,26,30,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	2,36,21,31,24,44
	Route 3	3,23,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39,44	3,32,5,35,40,44
$\|K\| = 4$	Obj value	2561.2	2561.2	2577.5	2813.8
	Route 1	1,32,5,35,18,44	1,32,5,35,18,44	1,32,9,16,5,44	1,32,5,18,13,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	36,31,24,41,44
	Route 3	3,23,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39,44
	Route 4	4,13,7,26,30,44	4,13,7,26,30,44	4,13,18,7,26,44	4,27,11,44
$\|K\| = 5$	Obj value	2151.0	2157.9	2183.2	2724.4
	Route 1	1,32,9,16,18,44	1,32,9,16,20,44	1,32,9,29,16,44	1,32,35,18,13,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44
	Route 3	3,42,15,39,10,44	3,42,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39,44
	Route 4	4,7,23,37,30,44	4,7,23,26,30,44	4,13,7,26,30,44	4,27,11,44
	Tour 5	5,35,40,13,26,44	5,35,40,18,13,44	5,28,20,35,18,44	5,14,44
$\|K\| = 6$	Obj value	1755.9	1871.0	1977.3	2120.8
	Route 1	1,32,9,16,20,44	1,32,9,16,23,44	1,32,9,29,16,44	1,32,9,29,16,44
	Route 2	2,36,31,24,41,44	2,36,19,21,44	2,36,31,24,41,44	2,36,19,21,44
	Route 3	3,42,15,39,10,44	3,42,37,10,39,44	3,42,23,37,10,44	3,23,37,10,39,44
	Route 4	4,7,23,37,30,44	4,13,31,24,41,44	4,13,7,26,30,44	4,13,7,26,30,44
	Route 5	5,35,18,13,26,44	5,18,7,26,30,44	5,20,35,18,44	5,20,25,34,35,44
	Route 6	6,25,34,27,40,44	6,25,20,35,40,44	6,28,25,34,44	6,11,12,44

Table 13
Results of 44 node case: robust model, $|K| = 2$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	3438.8	3439.3	3438.8	3457.6
	Route 1	1,3,23,37,10,44	1,7,4,26,30,44	1,3,23,37,10,44	1,32,5,35,40,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44
$\Gamma_k = 1$	Obj value	3452.5	3453.4	3452.5	3472.76
	Route 1	1;3,23,37,10,44	1,18;4,26,30,44	1;3,23,37,10,44	1,32,5,35,40;44
	Route 2	2,36;31,24,41,44	2,36; 31, 24,41,44	2,36;31,24,41,44	2,36;31,24,41,44
$\Gamma_k = 2$	Obj value	3460.7	3460.7	3460.7	3482.9
	Route 1	1;3,23,37,10;44	1;3,23,37,10;44	1;3,23,37,10;44	1,32;5,35,40;44
	Route 2	2,36;31,24,41;44	2,36;31,24,41;44	2,36;31,24,41;44	2,36;31,24,41;44
$\Gamma_k = 3$	Obj value	3465.7	3465.8	3465.7	3487.5
	Route 1	1;3;23,37,10;44	1;3;23,37,10;44	1;3;23,37,10;44	1,32;5,35;40;44
	Route 2	2,36;24;41,30;44	2,36;31,24;41;44	2,36;24;41,30;44	2,36;24;41,30;44
$\Gamma_k = 4$	Obj value	3468.5	3468.5	3468.5	3490.5
	Route 1	1;3;23;37,10;44	1;3;23;37,10;44	1;3;23;37,10;44	1;32;5,35;40;44
	Route 2	2,36;31;24;41;44	2,36;31;24;41;44	2,36;31;24;41;44	2,36;31;24;41;44
$\Gamma_k = 5$	Obj value	3469.5	3469.5	3469.5	3492.5
	Route 1	1;3;23;37;10;44	1;3;23;37;10;44	1;3;23;37;10;44	1;32;5;35;40;44
	Route 2	2;36;31;24;41;44	2;36;31;24;41;44	2;36;31;24;41;44	2;36;31;24;41;44

Table 14
Results of 44 node case: robust model, $|K| = 3$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	2995.4	2995.4	3011.2	3029.6
	Route 1	1,7,4,26,30,44	1,7,4,26,30,44	1,32,5,35,18,44	1,32,5,35,40,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	2,36,21,31,24,44
	Route 3	3,23,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39	3,23,37,10,39,44
$\Gamma_k = 1$	Obj value	3014.5	3014.5	3031.0	3049.5
	Route 1	1,18;4,26,30,44	1,18;4,26,30,44	1,32,5,35,18;44	1,32,5,35,40;44
	Route 2	2,36;31,24,41,44	2,36;31,24,41,44	2,36;31,24,41,44	2,36,21,31,24;44
	Route 3	3,23,37,10,39;44	3,23,37,10,39;44	3,23,37,10,39;44	3,23,37,10,39;44
$\Gamma_k = 2$	Obj value	3024.5	3024.5	3043.5	3064.0
	Route 1	1;18,13,26;30,44	1;18,13,26;30,44	1,32;5,35,18;44	1,32;5,35,40;44
	Route 2	2,36;31,24,41;44	2,36;31,24,41;44	2,36;31,24,41;44	2,36;21,31,24;44
	Route 3	3;23,37,10,39;44	3;23,37,10,39;44	3;23,37,10,39;44	3;23,37,10,39;44
$\Gamma_k = 3$	Obj value	3031.2	3031.2	3049.8	3071.9
	Route 1	1;18,13;26;30,44	1;18,13;26;30,44	1,32;5,35;18;44	1,32;5,35;40;44
	Route 2	2,36;31,24;41;44	2,36;31,24;41;44	2,36;24;41,30;44	2,36;21;31,24;44
	Route 3	3;23;37,10,39;44	3;23;37,10,39;44	3;23;37,10,39;44	3;23;37,10,39;44
$\Gamma_k = 4$	Obj value	3035.8	3035.8	3054.1	3076.1
	Route 1	1;18,13;26;30;44	1;18,13;26;30;44	1;32;5,35;18;44	1;32;5,35;40;44
	Route 2	2,36;31;24;41;44	2,36;31;24;41;44	2,36;31;24;41;44	2,36;21;31;24;44
	Route 3	3;23;37,10;39;44	3;23;37,10;39;44	3;23;37,10;39;44	3;23;37,10;39;44
$\Gamma_k = 5$	Obj value	3038.0	3038.0	3056.9	3078.9
	Route 1	1;18;13;26;30;44	1,18,13,26,30,44	1;32;5;35;18;44	1;32;5;35;40;44
	Route 2	2;36;31;24;41;44	2,36,31,24,41,44	2;36;31;24;41;44	2;36;21;31;24;44
	Route 3	3;23;37;10;39;44	3,23,37,10,39,44	3;23;37;10;39;44	3;23;37;10;39;44

Table 15
Results of 44 node case: robust model, $|K| = 4$.

		NoClus, Exact	NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	2561.2	2561.2	2668.7	2890.1
	Route 1	1,32,5,35,18,44	1,32,5,35,18,44	1,32,9,5,44	1,32,5,18,13,44
	Route 2	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44	2,36,31,24,41,44
	Route 3	3,23,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39,44	3,23,37,10,39,44
	Route 4	4,13,7,26,30,44	4,13,7,26,30,44	4,13,18,7,26,44	4,27,44
$\Gamma_k = 1$	Obj value	2585.1	2585.1	2696.7	2920.1
	Route 1	1,32;5,18,7,44	1,32;5,18,7,44	1,32,9,5;44	1,32,5,18,13;44
	Route 2	2,36;31,24,41,44	2,36;31,24,41,44	2,36;31,24,41,44	2,36;31,24,41,44
	Route 3	3,23,37,10,39;44	3,23,37,10,39;44	3,23,37,10,39;44	3,23,37,10,39;44
	Route 4	4;40,13,26,30,44	4;40,13,26,30,44	4,13,18;7,26,44	4,27;44
$\Gamma_k = 2$	Obj value	2600.1	2600.1	2714.6	2938.6
	Route 1	1,32;5,18;7,44	1,32,5,18,7,44	1,32,9;5;44	1,32;5,18,13;44
	Route 2	2,36;31,24,41;44	2,36,31,24,41,44	2,36;31,24,41;44	2,36;31,24,41;44
	Route 3	3;23,37,10,39;44	3,23,37,10,39,44	3;23,37,10,39;44	3;23,37,10,39;44
	Route 4	4;40,13,26;30,44	4,40,13,26,30,44	4,13,18;7,26;44	4;27;44
$\Gamma_k = 3$	Obj value	2611.1	2611.1	2722.2	2945.3
	Route 1	1,32;5,35;18;44	1,32;5,35;18;44	1;32,9;5;44	1,32;5;18,13;44
	Route 2	2,36;31,24;41;44	2,36;31,24;41;44	2,36;24;41,30;44	2,36;31,24;41;44
	Route 3	3;23;37,10,39;44	3;23;37,10,39;44	3;23;37,10,39;44	3;23;37,10,39;44
	Route 4	4,13;7,26;30;44	4,13;7,26;30;44	4,13,18;7;26;44	4;27;44
$\Gamma_k = 4$	Obj value	2616.7	2616.7	2727.2	2949.4
	Route 1	1;32;5,35;18;44	1;32;5,35;18;44	1;32;9;5;44	1;32;5;18,13;44
	Route 2	2,36;31;24;41;44	2,36;31;24;41;44	2,36;31;24;41;44	2,36;31;24;41;44
	Route 3	3;23;37,10;39;44	3;23;37,10;39;44	3;23;37,10;39;44	3;23;37,10;39;44
	Route 4	4,13;7;26;30;44	4,13;7;26;30;44	4;13,18;7;26;44	4;27;44
$\Gamma_k = 5$	Obj value	2620.8	2620.8	2729.3	2951.6
	Route 1	1;32;5;35;18;44	1;32;5;35;18;44	1;32;9;5	1;32;5;18;13;44
	Route 2	2;36;31;24;41;44	2;36;31;24;41;44	2;36;31;24;41;44	2;36;31;24;41;44
	Route 3	3;23;37;10;39;44	3;23;37;10;39;44	3;23;37;10;39;44	3;23;37;10;39;44
	Route 4	4;13;7;26;30;44	4;13;7;26;30;44	4;13;18;7;26;44	4;27;44

Table 16
Results of 101 node case: $|K| = 10$, $\Gamma_k = 0 - 2$.

		NoClus Heuristic	Greedy	*k*-means
$\Gamma_k = 0$	Obj value	5390.1	5482.3	6079.7
(Nominal)	Route 1	1,14,96,93,38,101	1,29,54,59,98,101	1,54,27,41,59,101
	Route 2	2,70,53,90,100,101	2,31,71,32,70,101	2,70,32,89,53,101
	Route 3	3,22,27,41,88,101	3,74,22,41,14,101	3,88,98,93,38,101
	Route 4	4,78,54,59,98,101	4,78,69,81,77,101	4,78,69,81,13,101
	Route 5	5,73,74,58,43,101	5,40,57,76,73,101	5,26,56,101
	Route 6	6,17,45,15,92,101	6,100,94,86,99,101	6,61,84,85,18,101
	Route 7	7,95,97,60,99,101	7,97,60,93,38,101	7,86,62,17,92,101
	Route 8	8,83,19,61,94,101	8,89,63,49,83,101	8,63,64,91,11,101
	Route 9	9,84,85,62,86,101	9,19,84,46,18,101	9,101
	Route 10	10,34,51,29,28,101	10,21,52,82,34,101	10,73,75,23,42,101
$\Gamma_k = 1$	Obj value	5415.3	5528.9	6134.9
	Route 1	1;14,96,93,38,101	1,29,54,59;98,101	1,54,27,41,59;101
	Route 2	2,70,53,90;100,101	2,31,71,32,70;101	2,70,32,89,53;101
	Route 3	3,22,27,41;88,101	3,74,22,41,14;101	3;88,98,93,38,101
	Route 4	4,77;54,59,98,101	4,78,69,81,77;101	4,78,69,81,13;101
	Route 5	5,73,74;58,43,101	5,40,57,76,73;101	5,26,56;101
	Route 6	6;17,45,15,92,101	6;100,94,86,99,101	6,61,19,84,85;101
	Route 7	7,95;97,60,99,101	7;97,60,93,38,101	7;86,62,17,92,101
	Route 8	8,83,19,61; 94, 101	8,83,49,89,53;101	8,63,11,12,20;101
	Route 9	9; 84, 85, 62, 86, 101	9,19,84,46,18;101	9,101
	Route 10	10, 34, 51, 29, 28; 101	10,21,52,82,34;101	10;73,75,23,42,101
$\Gamma_k = 2$	Obj value	5432.8	5546.5	6152.3
	Route 1	1;14;96,93,38,101	1,29,54,59;98;101	1,54;27,41,59;101
	Route 2	2,70;53,90;100,101	2;31,71,32,70;101	2,70,32,89;53;101
	Route 3	3;22,27,41;88,101	3,74,22,41;88;101	3;88;98,93,38,101
	Route 4	4,77;54,59;98,101	4,78,69,81;77;101	4,78,69,81;13;101
	Route 5	5;73,74;58,43,101	5;40,57,76,73;101	5;p26,56;101
	Route 6	6;17,45,15;92,101	6;100,94,86;92,101	6,61;19,84,85;101
	Route 7	7,95;97,60;99,101	7;97;60,93,38,101	7;86,62,17;92,101
	Route 8	8,83;19,61;94,101	8,83,49;89,53;101	8,63,11;12,20;101
	Route 9	9;84,85;62,86,101	9;19,84,46,18;101	9;101
	Route 10	10,34,51;29,28;101	10;21,52,82,34;101	10;73,75,23,42;101

Table 17
Results of 101 node case, $|K| = 10$, $\Gamma_k = 3 - 5$.

		NoClus Heuristic	Greedy	K-mean
$\Gamma_k = 3$	Obj value	5446.5	5559.9	6162.5
	Route 1	1;14;96;93,38,101	1,29;54,59;98;101	1,54;27;41,59;101
	Route 2	2,70;53,90;100;101	2;31,71,32;70;101	2,70;32,89;53;101
	Route 3	3;22;27,41;88,101	3;74,22,41;88;101	3;88;98;93,38,101
	Route 4	4,77;54,59;98;101	4,78;69,81;77;101	4,78;69,81;13;101
	Route 5	5;73,74;58,43;101	5;40;57,76,73;101	5;26;56;101
	Route 6	6;17;45,15;92,101	6;100,94,86;92;101	6,61;19;84,85;101
	Route 7	7;95;97,60;99,101	7;95;96;93,38,101	7;86;62,17;92,101
	Route 8	8,83;19;61;94,101	8;89,63;49,83;101	8;63,11;12,20;101
	Route 9	9;84;85;62,86,101	9;19,84;46,18;101	9;101
	Route 10	10;34,51;29,28;101	10;21;52,82,34;101	10;73,75,23;42;101
$\Gamma_k = 4$	Obj value	5452.2	5570.9	6169.6
	Route 1	1;14;96;93;38,101	1;29;54,59;98;101	1;54;27;41,59;101
	Route 2	2,70;28;90;100;101	2;31,71;32;70;101	2,70;32;89;53;101
	Route 3	3;22;27,41;88;101	3;74,22;41;14;101	3;88;98;93;38,101
	Route 4	4,78;77;59;98;101	4;78;69,81;77;101	4;78;69,81;13;101
	Route 5	5;73,74;58;43;101	5;40;57,76;73;101	5;26;56;101
	Route 6	6;17;45;15;92,101	6;100;94,86;92;101	6;61,84;85;18;101
	Route 7	7;95;97;60;99,101	7;97;60,93;38;101	7;86;62;17;92,101
	Route 8	8,83;19;61;94;101	8;89;63;49,83;101	8;63,11;12;20;101
	Route 9	9;84;85;62;86,101	9;19,84;46;18;101	9,101
	Route 10	10;34;51;29,54;101	10;21;52,82,34;101	10;73;75,23;42;101
$\Gamma_k = 5$	Obj value	5462.5	5579.1	6175.6
	Route 1	1;14;96;93;38;101	1;29;54;59;98;101	1;54;27;41;59;101
	Route 2	2;70;53;90;100;101	2;31;71;2;70;101	2;70;32;89;53;101
	Route 3	3;22;27;41;88;101	3;74;22;41;88;101	3;88;98;93;38;101
	Route 4	4;78;77;59;98;101	4;78;69;81;77;101	4;78;69;81;13;101
	Route 5	5;73;74;58;43;101	5;40;57;76;73;101	5;26;56;101

(*continued on next page*)

Table 17 (*continued*)

	NoClus Heuristic	Greedy	K-mean
Route 6	6;17;45;15;92;101	6;100;94;86;99;101	6;61;84;85;18;101
Route 7	7;95;97;60;99;101	7;95;96;93;38;101	7;86;62;17;92;101
Route 8	8;83;19;61;94;101	8;89;63;49;83;101	8;63;64;91;11;101
Route 9	9;84;85;62;86;101	9;19;84;46;18;101	9;101
Route 10	10;34;51;29;54;101	10;21;52;82;34;101	10;73;75;23;42;101

References

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

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research, 223*(2), 295–303.

Alonso-Mora, J., Samaranayake,S., Wallar, A., Frazzoli, E.. & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 201611675.

Armant, V. & Brown, K. N. (2014). Minimizing the driving distance in ride sharing systems. In *2014 IEEE 26th International Conference Tools with Artificial Intelligence (ICTAI)* (pp. 568–575). IEEE.

Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., & Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem. *Rapport de recherche- IMAG*.

Baldacci, R., Maniezzo, V., & Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research, 52*(3), 422–439.

Barreto, S., Ferreira, C., Paixao, J., & Santos, B. S. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research, 179*(3), 968–977.

Bistaffa, F., Farinelli, A., Ramchurn, S. D. (2014). *Sharing rides with friends: A coalition formation algorithm for ridesharing*.

Calvo, R. W., de Luigi, F., Haastrup, P., & Maniezzo, V. (2004). A distributed geographic information system for the daily car pooling problem. *Computers & Operations Research, 31*(13), 2263–2278.

Campbell, A. M., & Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science, 38*(3), 369–378.

Campbell, A. M., Vandenbussche, D., & Hermann, W. (2008). Routing for relief efforts. *Transportation Science, 42*(2), 127–145.

Cangialosi, E., Di Febbraro, A., & Sacco, N. (2016). Designing a multimodal generalised ride sharing system. *IET Intelligent Transport Systems, 10*(4), 227–236.

Chan, N. D., & Shaheen, S. A. (2012). Ridesharing in north america: Past, present, and future. *Transport Reviews, 32*(1), 93–112.

Deleplanque, S., Quilliot, A., & Bernay, B. (2014). Branch and price for a reliability oriented darp model. In *2014 International Conference Control, Decision and Information Technologies (CoDIT)* (pp. 081–086). IEEE.

Dondo, R., & Cerdá, J. (2007). A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research, 176*(3), 1478–1507.

Fanelli, A. & Greco, G. (2015). *Ride sharing with a vehicle of unlimited capacity*. arXiv preprint arXiv:1507.02414.

Ferguson, E. (1997). The rise and fall of the american carpool: 1970–1990. *Transportation, 24*(4), 349–376.

Fu, Y., Fang, Y., Jiang, C., & Cheng, J. (2008). Dynamic ride sharing community service on traffic information grid. In *2008 International Conference Intelligent Computation Technology and Automation (ICICTA)* (Vol. 2, pp. 348–352). IEEE.

Furuhata, M., Dessouky, M., Ordó nez, F., Brunet, M.-E., Wang, X., & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological, 57*, 28–46.

Herbawi, W. & Weber, M. (2012). The ridematching problem with time windows in dynamic ridesharing: a model and a genetic algorithm. In *2012 IEEE congress on evolutionary computation* (pp. 1–8). IEEE.

Hosni, H., Naoum-Sawaya, J., & Artail, H. (2014). The shared-taxi problem: Formulation and solution methods. *Transportation Research Part B: Methodological, 70*, 303–318.

Huang, Y., Bastani, F., Jin, R., & Wang, X. S. (2014). Large scale real-time ridesharing with service guarantee on road networks. *Proceedings of the VLDB Endowment, 7*(14), 2017–2028.

Jorgensen, R. M., Larsen, J., & Bergvinsdottir, K. B. (2007). Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society, 58*(10), 1321–1331.

Kelly, K. L. (2007). Casual carpooling-enhanced. Journal of Public. *Transportation, 10*(4), 6.

Lin, Y. and H. Shen (2016). Vshare: A wireless social network aided vehicle sharing system using hierarchical cloud architecture. In Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference, pp. 37–48. IEEE.

Manna, C. & Prestwich, S. (2014). Online stochastic planning for taxi and ridesharing. In *2014 IEEE 26th international conference on tools with artificial intelligence* (pp. 906–913). IEEE.

Mehrjerdi, Y. Z., & Nadizadeh, A. (2013). Using greedy clustering method to solve capacitated location-routing problem with fuzzy demands. *European Journal of Operational Research, 229*(1), 75–84.

Morency, C. (2007). The ambivalence of ridesharing. *Transportation, 34*(2), 239–253.

Nourinejad, M., & Roorda, M. J. (2016). Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies, 64*, 117–132.

Ordó nez, F. (2010). Robust vehicle routing. *TUTORIALS in Oprations Research*, 153–178.

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

Pinson, C., Afsar, H. M., & Prodhon, C. (2016). Heuristic approaches to solve a generalized dial-a-ride problem applied to car-pooling. *IFAC-PapersOnLine, 49*(12), 1187–1191.

Santos, D. O., & Xavier, E. C. (2015). Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications, 42*(19), 6728–6737.

Shen, W., Lopes, C. V., & Crandall, J. W. (2016). *An online mechanism for ridesharing in autonomous mobility-on-demand systems*. arXiv preprint arXiv:1603.02208.

Stiglic, M., Agatz, N., Savelsbergh, M., & Gradisar, M. (2016). Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review, 91*, 190–207.

Wang, X., Agatz, N., & Erera, A. (2017). Stable matching for dynamic ride-sharing systems. *Transportation Science, 52*(4).

Winter, S., & Nittel, S. (2006). Ad hoc shared-ride trip planning by mobile geosensor networks. *International Journal of Geographical Information Science, 20*(8), 899–916.

Yücenur, G. N., & Demirel, N.Ç. (2011). A new geometric shape-based genetic clustering algorithm for the multi-depot vehicle routing problem. *Expert Systems with Applications, 38*(9), 11859–11865.

Zhang, J., Wen, D., & Zeng, S. (2016). A discounted trade reduction mechanism for dynamic ridesharing pricing. *IEEE Transactions on Intelligent Transportation Systems, 17*(6), 1586–1595.