Integrated Vehicle Routing and Service Scheduling under Time and Cancellation Uncertainties with Application in Non-Emergency Medical Transportation

Xian Yu∗  Siqian Shen†  Huizhu Wang‡

Abstract

In this paper, we consider an integrated vehicle routing and service scheduling problem for serving customers in distributed locations who need pick-up, drop-off or delivery services. We take into account the random trip time, non-negligible service time and possible customer cancellations, under which an ill-designed schedule may lead to undesirable vehicle idleness and customer waiting. We build a stochastic mixed-integer program to minimize the operational cost plus expected penalty cost of customers’ waiting time, vehicles’ idleness and overtime. Furthermore, to handle real-time arrived service requests, we develop K-means clustering-based algorithms to dynamically update planned routes and schedules. The algorithms assign customers to vehicles based on similarities and then plan schedules on each vehicle separately. We conduct numerical experiments based on diverse instances generated from census data and data from the Ford Motor Company’s GoRide service, to evaluate result sensitivity and to compare the in-sample and out-of-sample performances of different approaches. Managerial insights are provided using numerical results based on different parameter choices and uncertainty settings.

Keywords: Pick-up and delivery with time windows; appointment scheduling; random cancellation; stochastic integer programming; K-means clustering

∗Department of Industrial and Operations Engineering, University of Michigan at Ann Arbor, USA; email: yuxian@umich.edu;
†Corresponding author; Department of Industrial and Operations Engineering, University of Michigan at Ann Arbor, USA; email: siqian@umich.edu;
‡Ford Motor Company; email: hwang152@ford.com
1 Introduction

Ford Motor Company’s Non-Emergency Medical Transportation (NEMT) provides reservation-based pick-up and drop-off services to patients who are elderly, disabled, or have chronic diseases, for traveling to medical requests (see Dickey, 2018). The NEMT type of businesses require a large fleet of vehicles to serve a large number of dispersed patients during peak hours, and these vehicles may be idle when the number of service requests is low. Some patients may cancel existing reservations, resulting in further system idleness. The trip time and service duration could also be random due to hourly traffic conditions and the difficulty of loading/unloading some patients, respectively. In NEMT, most patients are reported to wait for 10 to 20 minutes for their scheduled trips (see Bryant, 2019). Through better designed vehicle routes and schedules for NEMT, one can potentially reduce the total number of vehicles in operations and is then capable of covering new service regions only using the existing vehicles, while maintaining high vehicle utilization rates to attain financial profits and high quality of service.

Through popularizing NEMT types of systems, under-served populations having scarce mobility resources but high needs can potentially have reliable and affordable transportation means. Indeed, an NEMT-like system can be extended for medical home care delivery or grocery delivery, in which vehicles take certified nurses or medicines/goods to patients/customers rather than having them travel to hospitals/grocery stores. Amid the COVID-19 pandemic, this type of service is extremely important to self-quarantined COVID-19 patients with mild conditions and also to patients having chronic diseases who originally need to regularly visit their doctors and thus could have high cross-infection risk. The latter are also the most vulnerable population groups who have the highest fatality rate among all COVID-19 infected case due to their weak health conditions (see Centers for Disease Control and Prevention, 2020). Providing such a service can alleviate some stress on existing medical systems (e.g., hospitals, clinics and urgent care units) that need to focus on treating COVID-19 patients with critical conditions.

For goods, people and products delivery, existing models and approaches are mainly based on variants of the Vehicle Routing Problem (VRP) (Laporte, 1992, 2007). In a capacitated VRP (CVRP), multiple vehicles are dispatched from a single depot to meet all demand from customers and each vehicle seeks a feasible route within its capacity to deliver to or pick up from a set of customers before returning to the depot, so that the total traveling distance of all vehicles is minimized (see, e.g., Toth and Vigo, 2002; Fukasawa et al., 2006; Ralphs et al., 2003). The VRP
with time windows (VRPTW) studies problems where each customer should be served only within a specified time interval or time window (Bräysy and Gendreau, 2005a). In a more specific context, the pick-up and delivery problem with time windows (PDPTW) (see, e.g., Savelsbergh and Sol, 1995; Parragh et al., 2008) assumes that each request specifies the size of the load needed to be transported, the locations where it is to be picked up (the origins), and the locations where it is to be delivered (the destinations).

The settings of NEMT are the most relevant to PDPTW, except that we also take into account non-negligible service duration at patients’ locations, service cancellations, and the uncertainties. Moreover, we consider a hybrid case having both static and dynamic operations, such that the reservation-based system schedules a set of known service requests first, and then accommodates a few requests that may arrive on short notice in real-time operations. This needs to dynamically reschedule existing unfinished services and to compute the new schedule efficiently.

In this paper, we combine appointment scheduling models with PDPTW while taking into account several types of uncertainties. Two settings of problems are considered: i) in the static setting, we assume that all service requests are known before planning, based on which we make an initial schedule and route plan; ii) in the dynamic setting, customers arrive during real-time operations before their requested pick-up time. All the origin-destination (O-D) pairs and time windows are known at the time of reserving or announcing service requests; however, both the service duration and travel time could be stochastic. We also incorporate random cancellations such that customers who request services have a certain chance of not showing up.

We model the static problem as a two-stage stochastic mixed-integer linear program (TS-MILP) with the objective to minimize the total operational cost of dispatching and routing vehicles, plus the expected penalty cost of customer waiting, vehicle idleness and overtime for ensuring high quality of service. Specifically, in the first stage, we assign all customers to different vehicles and also make routing decisions for each vehicle. After observing random service duration/travel time/customers’ cancellations, we formulate a linear program to calculate customers’ waiting time, vehicles’ idle time and overtime. A rolling horizon method is proposed to extend the TS-MILP to solve dynamic vehicle routing and service scheduling for real-time operations. Solving the TS-MILP could suffer from incapability of attaining optimal solutions at scale. Notice that due to the non-zero service time and also uncertainties of multiple parameters, we cannot employ the traditional branch-and-price algorithm for VRP variants for solving the TS-MILP. To speed up computation for realistic problem sizes, we develop several heuristic approaches that are all based on the K-
means clustering algorithm (Jain, 2010). They first group all the customers into \( K \) clusters based on their O-D pair similarities, assign one vehicle to each cluster and then make a schedule for each vehicle separately. Some heuristics also perform swapping step after getting an initial clustering result based on customers’ time windows to distribute them more evenly to each vehicle. We are able to compute a real-world dataset from Ford as well as large-scale instances via the integration of the aforementioned optimization and data clustering techniques.

The remainder of the paper is organized as follows. In Section 2, we review the most relevant papers on variants of VRP and appointment scheduling. In Section 3, we describe stochastic optimization models of the static and dynamic problems. In Section 4, we develop clustering-based heuristics to improve the efficiency of the dynamic approach for serving large-scale regions. In Section 5, numerical studies are conducted using instances generated based on the real data of Ford Motor Company’s GoRide service. We demonstrate the efficacy of our approaches under diverse settings and reveal managerial insights for different uncertainty realizations. Section 6 concludes the paper and states future research directions.

2 Literature Review

Our problem is closely related to VRPTW (see, e.g., Desrochers et al., 1992; Bräysy and Gendreau, 2005a,b), Dial-a-Ride problem (DARP) (see, e.g., Cordeau and Laporte, 2007; Berbeglia et al., 2012), pick-up and delivery problem (PDP) (see, e.g., Savelsbergh and Sol, 1995; Berbeglia et al., 2010) and appointment scheduling problems (see, e.g., Gupta and Denton, 2008; Erdogan and Denton, 2013; Berg et al., 2014; Deng and Shen, 2016; Jiang et al., 2017). We refer to Laporte (1992) and Laporte (2007) for classical models for VRP and the related exact algorithms, classical heuristics, and metaheuristics. Cordeau and Laporte (2007) review the literature of DARP, demonstrate the main features of the problem and provide a summary of the most important models and algorithms. Savelsbergh and Sol (1995) distinguish PDP from standard VRP and present a survey of its models and solution approaches, with a primary focus on deterministic problems. Berbeglia et al. (2010) and Pillac et al. (2013) provide thorough reviews of dynamic PDP and VRP, respectively, where objects or people have to be served in real-time. Cömert et al. (2017) consider VRP with hard time windows (VRPHTW) using a “cluster-first route-second” hierarchical approach, where the authors first assign customers to vehicles using different clustering algorithms and then solve a VRPHTW as an MILP. Although the idea is very similar to our work, the authors do not take into account any
parameter uncertainty. In the context of dynamic and stochastic VRP, Bertsimas and Van Ryzin (1991), Bertsimas and Van Ryzin (1993) and Bertsimas and Simchi-Levi (1996) develop and review the greedy heuristics for solving VRP with stochastic dynamically arriving demand, mainly based on queueing theories. On the other hand, Dror et al. (1989) review the stochastic programming models for VRPTW with stochastic demand and propose a new solution frame using Markov Decision Process (MDP). Powell (1996), Powell et al. (2000), Simao et al. (2009) apply the MDP to truckload assignment problem and develop a deterministic myopic method, a stochastic dynamic model and Approximate Dynamic Programming for estimating future cost functions, respectively. Bent and Van Hentenryck (2004) consider online stochastic multiple vehicle routing with time windows in which customers arrive dynamically and the goal is to maximize the number of customers served. The authors propose a multiple scenario approach (MSA) that continuously generates routing plans for scenarios including known and future requests. Later, Bent and Van Hentenryck (2007) propose to include customer waiting and relocation in the online algorithm to achieve better results. More recently, Bertsimas et al. (2019) specifically consider online vehicle routing solved by ride-sharing companies and propose an optimization framework and an efficient algorithm to allow solving the problem on demand at a large scale, demonstrated by numerical results using real New York City taxi data.


Recently, Jiang et al. (2017) consider a single-server appointment scheduling problem with random no-shows and service duration. They derive mixed-integer nonlinear programming reformulations, valid inequalities and convex-hull representations under specially structured ambiguity sets for distributionally robust appointment scheduling. Deng and Shen (2016) investigate multi-server scheduling problem with random service duration and minimize the cost of operating servers, subject to a joint chance constraint limiting the risk of server running overtime.
For home health care (HHC) routing and scheduling, Fikar and Hirsch (2017) present a comprehensive review with a focus on the various problem settings and solution approaches. Among them, Heching et al. (2019) use a logic-based Benders' decomposition (LBBD) to solve the assignment-scheduling problem and propose several subproblem relaxations to speed up the computation. Most literature consider static information, i.e., all data are known in advance and no uncertainty in the parameter is considered. Several papers deal with parameter uncertainties and among them, Lanzarone and Matta (2014) and Carello and Lanzarone (2014) consider demand uncertainty. They formulate the nurse-to-patient assignment problem as a robust optimization model and propose both analytical and heuristic-based approaches. Yuan et al. (2015) study random service time and propose Column Generation (CG) and several heuristics to solve a stochastic program for optimizing routing and scheduling decisions. The random service time is also considered in Zhan and Wan (2018); Zhan et al. (2021), where the former formulates a Scenario-based Mixed-Integer Program (SBMIP) and develops an algorithm based on Tabu Search to efficiently solve the problem, and the latter proposes an L-shaped method with valid inequalities to speed up the solution process. An easy-to-implement heuristic based on a modified Traveling Salesman Problem (MTSP) is also developed in Zhan et al. (2021) for solving large-scale instances. We compare our work with the literature in Table 1, where we also present the maximum size of instances that each paper solves in the last two columns with $|J|$ and $|I|$ being the number of vehicles and customers, respectively.

**Main Contributions of This Paper.** To our best knowledge, this paper is the first to combine PDPTW with appointment scheduling and incorporates various uncertainty sources, including time-related uncertainty and cancelations. Our stochastic optimization model comprehensively captures a wide spectrum of decisions made in related applications, ranging from vehicle routing, service sequencing, to specific vehicle arrival and departure time for appointment scheduling. We decompose the model into two stages, such that the second-stage problem can efficiently compute waiting time, idle time and overtime of an optimal schedule for each vehicle via a linear programming model. Moreover, using the clustering-based heuristics, we are able to quickly compute high-quality solutions for large-scale instances with 40 vehicles, 450 demand requests, within several seconds.

### 3 Optimization Models

We first consider a reservation-based transportation system, where customers need to provide trip information including O-D pairs and pick-up time windows when reserving their trips. After the
Table 1: Comparison between our study and HHC routing and scheduling literature

| Papers                      | Problem description          | Stochasticity         | Dynamic | Exact   | Heuristics                      | $|J|$ | $|t|$ |
|-----------------------------|-----------------------------|-----------------------|---------|---------|-------------------------------|------|------|
| Our work                   | PDPTW + daily planning      | random travel time/   | rolling | MILP    | K-means clustering            | 40   | 450 p/d* |
|                             |                             | service duration/     |         |         |                               |      |      |
|                             |                             | customers' cancellations |         |         |                               |      |      |
| Zhan and Wan (2018)         | VRP + daily planning        | service duration      | -       | SBMIP   | Tabu Search                   | 7    | 40 p/d* |
|                             |                             |                       |         |         |                               |      |      |
| Zhan et al. (2021)          | VRP + daily planning        | service duration      | -       | L-shaped MTSP |                      | 1    | 10 p/d* |
| Heching et al. (2019)       | VRPTW + weekly planning     | -                     | rolling | LBBBD   | subproblem relaxation         | 20   | 60 p/w*|
| Allaoua et al. (2013)       | VRPTW + daily planning      | -                     | -       | ILP*    | set partitioning              | 9    | 30 p/d* |
| Rasmussen et al. (2012)     | VRPTW + daily planning      | -                     | -       | B&P*    | clustering                    | 15   | 150 p/d*|
| Lanzarone and Matta (2014)  | nurse-to-patient assignment | random demand         | rolling | analytical policy based on sorting | 8    | 40 p/w* |
|                             |                             | + robust optimization |         |         |                               |      |      |
| Bennett and Erera (2011)    | VRP + weekly planning       | -                     | rolling | -       | distance-based insertion,     | 1    | 1.5 p/d*|
|                             |                             |                       |         |         | capacity-based insertion      |      |      |
| Yuan et al. (2015)          | VRPTW + daily planning      | random service times  | -       | CG      | greedy heuristic, variable neighbourhood descent algorithm | 60   | 50 p/d* |
|                             |                             | + stochastic program   |         |         |                               |      |      |
| Nickel et al. (2012)        | VRPTW + weekly planning     | -                     | greedy-based insertion | -       | constraint programming, heuristic + adaptive large neighborhood search | 12   | 95 p/w* |
| Cappanera and Scutellà (2015)| VRPTW + weekly planning    | -                     | -       | ILP*    |                               | 11   | 162 p/w*|
| Bard et al. (2014)          | VRPTW + weekly planning     | -                     | -       | B&P&C*  | rolling horizon method        | 20   | 650 p/w*|

* ILP: integer linear program; B&P: branch-and-price; B&P&C: branch-and-price-and-cut; p/d: per day; p/w: per week

The operating coordinator gathers the information of all the trips, he or she needs to assign them to a fleet of vehicles and also make an initial schedule while considering the uncertainty in the service duration/travel time/customers’ cancellations. We illustrate the problem in Figure 1, where we have one depot (denoted by a square), three vehicles (denoted by different colored lines), and seven customers with O-D pairs $(O_i, D_i), i = 1, \ldots, 7$ (denoted by black dots). The arrow lines indicate an optimal solution of vehicle routing decisions, based on which we can make corresponding scheduling decisions.

Figure 1: A single-depot vehicle routing and appointment scheduling problem.
All the time-related notations \((t \text{ and } T)\) marked on the figure are explained later in Section 3.1 and modeled by random variables. Moreover, each customer who made a reservation may not show up with a certain chance, which we can also model using random variables having 0 or 1 realized values. Next, we define our notation in Section 3.1, and present formulations of static and dynamic problems in Sections 3.2 and 3.3, respectively.

### 3.1 Problem description and notation

We use \(I, J, K\) to denote the sets of customers, vehicles and service slots in each vehicle, respectively. (Each service slot can only fulfill one request and a slot can be used only when we have used up all the earlier slots in the same vehicle.) For notation simplicity, we assume that all vehicles have the same number of slots, \(|K|\), as the maximum number of service requests a vehicle can serve within the operational time frame. Note that this assumption is made without loss of generality, as if no request is assigned to any remaining slots on a vehicle, the vehicle returns to the depot and therefore the depot will occupy all the remaining slots. Let \(O_i, D_i\) be the origin and destination of the trip requested by customer \(i\), \([a_i, \bar{a}_i]\) be the requested pick-up time window of customer \(i\), \(L_j\) and \(c_j\) be the total operating time and operational cost of vehicle \(j\) for all \(i \in I\) and \(j \in J\). For example, if the operational time frame is between 4am and 7pm, we set \(L_j = 60 \text{ minutes per hour } \times 15 \text{ hours } = 900 \text{ minutes}\). Denote \(c^w\), \(c^n\), \(c^o\) as the unit penalty costs of customers’ waiting time, vehicles’ idle time, and overtime, respectively. (For notation simplicity, we assume that these costs are the same across all customers or vehicles. They can easily be differentiated for individual customers or vehicles without affecting our models later.) Let \(N\) be the set of locations, containing all the origins, destinations and the depot, i.e., \(N = \{O_i, D_i\}_{i \in I} \cup \{\text{depot}\}\).

Parameter \(\xi\) denotes the overall vector of uncertain parameters and let \(P\) be its probability distribution. Without loss of generality, we consider discrete distribution \(P\) and a finite set of realizations of the random vector \(\xi\). In practice, if the true distribution \(P\) of random variable \(\xi\) is continuous, we apply the Monte Carlo sampling approach to replace \(P\) with an empirical distribution constructed by \(|\Omega|\) scenarios with each scenario \(\omega \in \Omega\) having an equal probability \(p^\omega = 1/|\Omega|\). The resulting problem with the constructed scenarios is called the Sample Average Approximation (SAA) problem (see, Kleywegt et al., 2002). Specifically, for each scenario \(\omega \in \Omega\), we denote \(\tau_{n_1,n_2}(\omega)\) as the travel time between \(n_1\) and \(n_2\) for all \(n_1, n_2 \in N\), \(\hat{T}_{Oi}(\omega)\), \(\hat{T}_{Di}(\omega)\) as the service duration at customer \(i\)’s origin and destination, respectively, and \(T_i(\omega)\) as the total time for serving customer \(i\), where \(T_i(\omega) = \hat{T}_{Oi}(\omega) + \tau_{O_i,D_i}(\omega) + \hat{T}_{Di}(\omega)\). Also, for all \(i_1, i_2 \in I\), let \(t_{i_1,i_2}(\omega)\)
be the transition time from customer $i_1$ to customer $i_2$, and for all $i \in I$, let $t_{i, \text{depot}}(\omega)$, $t_{\text{depot}, i}(\omega)$ be the transition time from the depot to customer $i$ and from customer $i$ to the depot, respectively. Finally, we consider $q_i(\omega)$ as a random service-cancellation outcome of customer $i$, which equals to 1 if customer $i$ shows up when an assigned vehicle arrives, and 0 otherwise, for all $i \in I$. The overall random vector is $\xi = (T, \hat{T}, \tau, t, q)$. (Throughout the paper, we use bold symbols to denote the vector form of a decision variable or a parameter.)

We define binary variables $x_j$, $y_{ij}$, $u^j_{i,k}$, $u^j_{\text{depot}, k} \in \{0, 1\}$ for all $i \in I$, $j \in J$, $k \in K$ such that $x_j = 1$ if we operate vehicle $j$, $y_{ij} = 1$ if we assign customer $i$ to vehicle $j$, $u^j_{i,k} = 1$ if customer $i$ is assigned to the $k^{th}$ slot of vehicle $j$, and $u^j_{\text{depot}, k} = 1$ if the depot is assigned to the $k^{th}$ slot of vehicle $j$, respectively. We also define continuous variables $r^j_k \geq 0$ for all $k \in K$, $j \in J$ as the planned start time of the $k^{th}$ request on vehicle $j$. Then, in the second stage, for each scenario $\omega \in \Omega$, variables $w^j_k(\omega)$ represent the customer’s waiting time at the beginning of the $k^{th}$ slot on vehicle $j$, $l^j_k(\omega)$ the vehicle’s idle time at the end of the $k^{th}$ slot on vehicle $j$, $l^j_0(\omega)$ the vehicle’s idle time before the first request starts on vehicle $j$, $W^j(\omega)$ the overtime of vehicle $j$, and $V^j(\omega)$ the total travel time of vehicle $j$ for all $j \in J$, $k \in K$.

![Figure 2](image_url)

**Figure 2**: Relationship between planned schedule and realized waiting time, idle time, overtime depending on observed time duration in one scenario for a vehicle.

Figure 2 depicts the relationship between the decision variables and parameters. Given a planned schedule ($r^j_k$) with observed service duration ($T_i(\omega)$) and transition time ($t_{i_1, i_2}(\omega)$), one can easily calculate the waiting ($w^j_k(\omega)$), idling ($l^j_k(\omega)$), and overtime ($W^j(\omega)$) for each scenario $\omega$ based on this figure. We formalize it in (2a)–(2f), which can be transformed to a set of linear constraints.
3.2 Static vehicle routing and service scheduling

We formulate a TS-MILP to model the vehicle routing and service scheduling problem, where the first-stage problem decides which vehicles to operate \((x_j)\), the assignment of each customer to vehicles \((y_{ij})\), the relative slots in each vehicle \((u_{i,k}^j)\), and the planned start time on each vehicle \((r_{jk}^j)\).

Let \(Q(u, r, \omega)\) be the total penalty cost of waiting, idling and overtime given first-stage decision variables \(u\), \(r\) and the realization of uncertainty \(\xi\) in scenario \(\omega\). Then, the SAA reformulation can be represented as follows.

\[
\begin{align*}
\min & \quad \sum_{j \in J} c_j x_j + \sum_{\omega \in \Omega} p^\omega Q(u, r, \omega) \\
\text{s.t.} & \quad \sum_{j \in J} y_{ij} = 1, \quad \forall i \in I, \quad (1a) \\
& \quad y_{ij} \leq x_j, \quad \forall i \in I, \ j \in J, \quad (1b) \\
& \quad \sum_{k \in K} u_{i,k}^j = y_{ij}, \quad \forall i \in I, \ j \in J, \quad (1c) \\
& \quad \sum_{i \in I} u_{i,k}^j + u_{\text{depot},k}^j = x_j, \quad \forall k \in K, \ j \in J, \quad (1d) \\
& \quad \sum_{i \in I} u_{i,k+1}^j \leq \sum_{i \in I} u_{i,k}^j, \quad \forall k = 1, \ldots, |K| - 1, \ j \in J, \quad (1e) \\
& \quad \sum_{i \in I} a_i u_{i,k}^j + \left(1 - \sum_{i \in I} u_{i,k}^j\right) L_j \leq r_{k}^j \leq \sum_{i \in I} \bar{a}_i u_{i,k}^j + \left(1 - \sum_{i \in I} u_{i,k}^j\right) L_j, \quad \forall k \in K, \ j \in J, \quad (1f) \\
& \quad r_{k+1}^j \geq r_{k}^j, \quad \forall k = 1, \ldots, |K| - 1, \ j \in J, \quad (1g) \\
& \quad x_j \in \{0, 1\}, \ y_{ij} \in \{0, 1\}, \ u_{i,k}^j \in \{0, 1\}, \ r_{k}^j \geq 0, \ \forall i \in I, \ k \in K, \ j \in J. \quad (1i)
\end{align*}
\]

The objective \((1a)\) minimizes the total operational cost and an expected second-stage cost \(Q(u, r, \omega)\).

Constraints \((1b)\)–\((1d)\) ensure that every request is assigned to a slot on an operating vehicle. Constraints \((1e)\) assign at most one request to each slot on each vehicle, and if there are no requests assigned to this slot, the vehicle returns to the depot. Constraints \((1f)\) prohibit assigning a request to a slot on a vehicle if an earlier slot is vacant. Constraints \((1g)\) ensure that each request on vehicle \(j\) starts within its requested time window, and if no requests are assigned to vehicle \(j\) at the \(k^{th}\) slot, \(r_{k}^j\) is set as the time limit \(L_j\) for vehicle \(j\). Constraints \((1h)\) ensure the requests’ start time being in line with their order.

In the second stage under scenario \(\omega\), the waiting time \(w_{1}^j(\omega), \ w_{k+1}^j(\omega)\), idle time \(l_{0}^j(\omega), \ l_{k}^j(\omega)\).
and overtime \( W_j(\omega) \) for all \( k = 1, \ldots, |K| - 1 \) and \( j \in J \) can be respectively measured by

\[
\begin{align*}
    w^i_1(\omega) &= \max \left\{ 0, \sum_{i \in I} t_{\text{depot},i}(\omega) u^i_{1,k} - r^i_1 \right\}, \quad (2a) \\
    l^0_0(\omega) &= \max \left\{ 0, r^i_1 - \sum_{i \in I} t_{\text{depot},i}(\omega) u^i_{1,k} \right\}, \quad (2b) \\
    w^j_{k+1}(\omega) &= \max \left\{ 0, r^j_k + w^j_k(\omega) + \sum_{i \in I} q_i(\omega) T_i(\omega) u^i_{k,k} + \sum_{i, i_2 \in I \cup \{\text{depot}\}} t_{i,i_2}(\omega) u^i_{1,k} u^i_{i_2,k+1} - r^j_{k+1} \right\}, \quad (2c) \\
    l^j_k(\omega) &= \max \left\{ 0, r^j_{k+1} - \left( r^j_k + w^j_k(\omega) + \sum_{i \in I} q_i(\omega) T_i(\omega) u^i_{k,k} + \sum_{i, i_2 \in I \cup \{\text{depot}\}} t_{i,i_2}(\omega) u^i_{1,k} u^i_{i_2,k+1} \right) \right\}, \quad (2d) \\
    W_j(\omega) &= \max \left\{ 0, r^j_{|K|} + w^j_{|K|}(\omega) + \sum_{i \in I} q_i(\omega) T_i(\omega) u^i_{|K|,k} + \sum_{i, i_2 \in I \cup \{\text{depot}\}} t_{i,i_2}(\omega) u^i_{1,k} u^i_{i_2,k+1} - L_j \right\}, \quad (2e) \\
    l^j_{|K|}(\omega) &= \max \left\{ 0, L_j - \left( r^j_{|K|} + w^j_{|K|}(\omega) + \sum_{i \in I} q_i(\omega) T_i(\omega) u^i_{|K|,k} + \sum_{i, i_2 \in I \cup \{\text{depot}\}} t_{i,i_2}(\omega) u^i_{1,k} u^i_{i_2,k+1} \right) \right\}. \quad (2f)
\end{align*}
\]

In (2a)–(2f), the actual transition time \( t_{i,i_2}(\omega) \) between customer \( i_1 \) and customer \( i_2 \) depends on the service cancellation probability of customer \( i_1 \), i.e., if \( i_1 \) cancels the service, then the vehicle will travel from \( i_1 \)'s origin directly; otherwise, the vehicle travels from \( i_1 \)'s destination to \( i_2 \)'s origin. The calculations of the random travel time are given by

\[
t_{i_1,i_2}(\omega) = q_{i_1}(\omega) \tau_{D_{i_1},O_{i_2}}(\omega) + (1 - q_{i_1}(\omega)) \tau_{O_{i_1},D_{i_2}}(\omega),
\]

for all \( i_1, i_2 \in I \). Specially, for each \( i \in I \), we have

\[
t_{\text{depot},i}(\omega) = \tau_{\text{depot},O_i}(\omega),
\]

\[
t_{i,\text{depot}}(\omega) = q_i(\omega) \tau_{D_i,\text{depot}}(\omega) + (1 - q_i(\omega)) \tau_{O_i,\text{depot}}(\omega).
\]

We illustrate different cases of waiting and idle time in Figure 3, where we assume that on vehicle \( j \), the \( k \)th and \((k + 1)\)th slots are assigned to customers \( i_1 \) and \( i_2 \), respectively. The upper figure shows the scenario when we have idle time at the end of \( k \)th slot, while the lower figure shows the scenario when we have waiting time at the beginning of \((k + 1)\)th slot. Note that the differences between \( w^j_{k+1}(\omega) \) and \( l^j_k(\omega) \) for all \( k = 0, \ldots, |K| - 1 \) and the difference between \( W_j(\omega) \) and \( l^j_{|K|}(\omega) \) are always constants. That is, in scenario \( \omega_1 \) (see upper figure in Figure 3), we have

\[
w^j_{k+1}(\omega) - l^j_k(\omega) = -l^j_k(\omega) = r^j_k + w^j_k(\omega) + T_{i_1}(\omega) + t_{i_1,i_2}(\omega) - r^j_{k+1},
\]

and in scenario \( \omega_2 \) (see lower figure in Figure 3), we have

\[
w^j_{k+1}(\omega) - l^j_k(\omega) = w^j_{k+1}(\omega) = r^j_k + w^j_k(\omega) + T_{i_1}(\omega) + t_{i_1,i_2}(\omega) - r^j_{k+1},
\]

11
Figure 3: Illustration of waiting and idle time of slots $k$ and $k + 1$ on vehicle $j$. 

<table>
<thead>
<tr>
<th>Scenario $\omega_1$:</th>
<th>Customer $i_1$</th>
<th>Customer $i_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{k+1}^j(\omega_1) = 0$</td>
<td>$w_k^j(\omega_1)$</td>
<td>$T_i(\omega_1)$</td>
</tr>
<tr>
<td>$l_k^j(\omega_1) \geq 0$</td>
<td>$t_{i_1,i_2}(\omega_1)$</td>
<td>$l_{i_2}^j(\omega_1)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario $\omega_2$:</th>
<th>Customer $i_1$</th>
<th>Customer $i_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{k+1}^j(\omega_2) = 0$</td>
<td>$w_k^j(\omega_2)$</td>
<td>$T_i(\omega_2)$</td>
</tr>
<tr>
<td>$l_k^j(\omega_2) \geq 0$</td>
<td>$t_{i_1,i_2}(\omega_2)$</td>
<td>$w_{k+1}^j(\omega_2)$</td>
</tr>
</tbody>
</table>

For a given VRP solution $u$, $r$ and scenario $\omega \in \Omega$, we formulate $Q(u, r, \omega) =$

\[
\min \sum_{j \in J} \left( \sum_{k \in K} \left( c^w w_k^j(\omega) + c^l l_k^j(\omega) \right) + c^w W_j(\omega) + c^d V_j(\omega) \right) \\
\text{s.t. } w_k^j(\omega) - l_k^j(\omega) = \sum_{i \in I} t_{\text{depot},i}(\omega) u_{i,k} - r_k^j, \forall j \in J, \tag{3a}
\]

\[
w_k^j(\omega) - l_k^j(\omega) = \sum_{i \in I} q_i(\omega) T_i(\omega) u_{i,k}^j + \sum_{i_1,i_2 \in I \cup \{\text{depot}\}} t_{i_1,i_2}(\omega) u_{i_1,k}^j u_{i_2,k+1}^j - r_k^j, \forall k = 1, \ldots, |K| - 1, \ j \in J, \tag{3b}
\]

\[
W_j(\omega) - l_k^j(\omega) - u_k^j(\omega) = r_k^j + \sum_{i \in I} q_i(\omega) T_i(\omega) u_{i,k}^j + \sum_{i_1,i_2 \in I \cup \{\text{depot}\}} t_{i_1,i_2}(\omega) u_{i_1,k}^j u_{i_2,k+1}^j - r_k^j, \forall k = 1, \ldots, |K| - 1, \ j \in J, \tag{3c}
\]

\[
W_j(\omega) = \sum_{k \in K} \sum_{i \in I} q_i(\omega) T_i(\omega) u_{i,k}^j + \sum_{i \in I} t_{\text{depot},i}(\omega) u_{i,1}^j + \sum_{k=1}^{|K|-1} \sum_{i_1,i_2 \in I \cup \{\text{depot}\}} t_{i_1,i_2}(\omega) u_{i_1,k}^j u_{i_2,k+1}^j + \sum_{i \in I} t_{i,\text{depot}}(\omega) u_{i,K}^j, \forall j \in J, \tag{3d}
\]

\[
w_k^j(\omega) \geq 0, \ l_k^j(\omega) \geq 0, \ l_k^j(\omega) \geq 0, \ W_j(\omega) \geq 0, \ V_j(\omega) \geq 0, \forall k \in K. \tag{3e}
\]

The objective function (3a) minimizes the total penalty of waiting, idleness, overtime and total travel time in scenario $\omega$. Constraints (3b) and (3c) yield either the waiting time of the $(k + 1)^{th}$ slot or the vehicle’s idle time after finishing the $k^{th}$ slot, both of which will have the values as in (2a)–(2d). Similarly, Constraints (3d) yield either the overtime $W_j(\omega)$ or the idle time $l_k^j(\omega)$.

Constraints (3e) calculate the total travel time of each vehicle by summing over the service time in each slot, and the travel time between any two adjacent slots. All the waiting, idleness, and
overtime variables are non-negative according to Constraints (3f).

The second-stage value function $Q(u, r, \omega)$ is a non-convex function with respect to $u$ because of the bilinear term $u_{i_1,k}^j u_{i_2,k+1}^j$ on the right-hand-side of (3c). Given binary-valued $u_{i_1,k}^j$ and $u_{i_2,k+1}^j$, we provide exact reformulations of the bilinear terms $z_{i_1,i_2,k}^j = u_{i_1,k}^j u_{i_2,k+1}^j$ in (3c) using McCormick envelopes:

\begin{align}
    z_{i_1,i_2,k}^j &\leq u_{i_1,k}^j, \quad \forall i_1, i_2 \in I \cup \{\text{depot}\}, \quad k = 1, \ldots, |K| - 1, \ j \in J, \quad (4a) \\
    z_{i_1,i_2,k}^j &\leq u_{i_2,k+1}^j, \quad \forall i_1, i_2 \in I \cup \{\text{depot}\}, \quad k = 1, \ldots, |K| - 1, \ j \in J, \quad (4b) \\
    z_{i_1,i_2,k}^j &\geq u_{i_1,k}^j + u_{i_2,k+1}^j - 1, \quad \forall i_1, i_2 \in I \cup \{\text{depot}\}, \quad k = 1, \ldots, |K| - 1, \ j \in J. \quad (4c)
\end{align}

We add variables $z_{i_1,i_2,k}^j$, $\forall i_1, i_2 \in I \cup \{\text{depot}\}$, $k = 1, \ldots, |K| - 1, \ j \in J$ and Constraints (4a)–(4c) into the first-stage problem. As a result, the second-stage value function is now a convex function in terms of the first-stage decisions $u, r, z$, and we denote it as $Q(u, r, z, \omega)$ to replace the original $Q(u, r, \omega)$ in Model (1).

In Appendix A, we provide modeling details of three extensions to Model (1) for accommodating various practical issues, including allowing ride-pooling and multiple customers sharing one ride (Extension I), enforcing deadlines for dropping off customers (Extension II), and allowing vehicles dispatched from multiple depots (Extension III).

### 3.3 Dynamic vehicle routing and service scheduling

Model (1) provides initial routes and schedules for each vehicle on a day-to-day basis. However, in the NEMT application, trip schedulers often observe that customers request trips in a short notice. A common mechanism for handling dynamic demand arrivals is to use a rolling-horizon-based approach, in which plans are made using all known information within a planning horizon, but decisions are not finalized until necessitated by a deadline. At each execution of the algorithm, the planning horizon is “rolled” forward to include more information, and we resolve the problem and implement some decisions with updated input data and parameters. Next, we elaborate how to extend our models in a rolling horizon framework to handle real-time service requests.

We can optimize Model (1) with updated parameters each time when a new request becomes known, and assume that no new service request will be considered while executing the algorithm. Specifically, when a new request shows up at time $s$, we assume the following sequence of events. First, we update all vehicles’ current status, including the time when they become available, $\hat{r}_s^j$, and the corresponding locations when they become available, $\hat{O}_s^j$, for all $j \in J$. There are five possible
Figure 4: Illustration of rolling horizon timings.

cases when a new request can occur (i.e., time $s$) and we specify their corresponding $(\hat{r}_s^j, \hat{O}_s^j)$-values in Figure 4 and as follows.

- **Cases 1 and 2:** when the assigned customer $i_1$ is waiting for vehicle $j$ or has already boarded, we let the vehicle finish its current request and then become available. In these two cases, we set $\hat{r}_s^j = r_k^j + w_k^j(\omega_1) + q_{i_1}(\omega_1)T_{i_1}(\omega_1)$, $\hat{O}_s^j = D_{i_1}$ if customer $i_1$ does not cancel the reservation (i.e., $q_{i_1}(\omega_1) = 1$), and $\hat{O}_s^j = O_{i_1}$ otherwise;

- **Case 3:** when the vehicle is traveling from the previous customer’s destination to the next customer’s origin and $l_k^j(\omega_1) \geq 0$, we set $\hat{r}_s^j = r_k^j + w_k^j(\omega_1) + q_{i_1}(\omega_1)T_{i_1}(\omega_1) + t_{i_1,i_2}(\omega_1)$ and $\hat{O}_s^j = O_{i_2};$

- **Case 4:** when the vehicle is idle, we set $\hat{r}_s^j = s$ and $\hat{O}_s^j = O_{i_2};$

- **Case 5:** when the vehicle is traveling from last customer’s destination to next customer’s origin and $w_{k+1}^j(\omega_2) \geq 0$, we let the vehicle finish the $(k + 1)^{th}$ slot’s request and set $\hat{r}_s^j = r_{k+1}^j + w_{k+1}^j(\omega_2) + q_{i_2}(\omega_2)T_{i_2}(\omega_2)$, $\hat{O}_s^j = D_{i_2}$ if customer $i_2$ does not cancel the reservation (i.e., $q_{i_2}(\omega_2) = 1$), and $\hat{O}_s^j = O_{i_2}$ otherwise.

Let $I_s$ be the set of all service requests announced prior to time $s$, excluding the ones that have been either completed or started (e.g., customers are waiting or on board). We then re-optimize the vehicle-customer assignments and corresponding schedules by solving a two-stage stochastic programming model similar to Model (1), where the only differences are that we replace all $I$ with $I_s$ and drop the binary variables $x_i$, meaning that all currently operating vehicles will be in use.
The second-stage problem is also similar to Model (3), except that we replace all $I$ with $I_s$ and replace Constraints (3b) with

$$w^j_1(\omega) - l^j_1(\omega) = \hat{r}^j_s + \sum_{i \in I} t_{\hat{O}^j_s, O_i}^j(\omega) w^j_1 - r^j_{1}, \forall j \in J,$$

because vehicle $j$ will start its service at the available time $\hat{r}^j_s$ and the origin of it now becomes $\hat{O}^j_s$, rather than the depot.

We summarize the detailed steps of rolling horizon method based on optimization models for dynamic routing and scheduling in Algorithm 1.

**Algorithm 1** Rolling horizon method based on optimization models

1: Solve the initial scheduling-routing problem (1) and obtain an optimal schedule and routing plan $(\bar{u}, \bar{r})$.
2: Sample one out-of-sample scenario $\omega$ and implement $\bar{u}, \bar{r}$ based on scenario $\omega$.
3: while a new request $i^*$ shows up at time $s$ do
4: Initialize $I_s = \emptyset$.
5: for each customer $i$ in $I$ do
6: if the announce time $s$ is earlier than the planned start time of $i$ in $\bar{r}$ then
7: Put customer $i$ into $I_s$.
8: else
9: Customer $i$ has been served.
10: end if
11: end for
12: Put customer $i^*$ into $I_s$ and update $I = I_s$.
13: Gather all vehicles’ status $(\hat{r}^j_s, \hat{O}^j_s)$ according to Cases 1-5 listed above.
14: Solve a variant of Model (1) with input $(\hat{r}^j_s, \hat{O}^j_s, I)$ and obtain an optimal schedule and routing plan $(\bar{u}, \bar{r})$.
15: Sample one out-of-sample scenario $\omega$ and implement $\bar{u}, \bar{r}$ based on scenario $\omega$.
16: end while

Notice that Algorithm 1 is not restricted to optimization-based models. In fact, as long as there is a way to update scheduling and routing plans, we can always apply the above rolling horizon method. For example, we can combine Algorithm 1 with clustering-based heuristics, which we will introduce next.

4 Two-phase Heuristics using Data Clustering

While we can attain solution optimality by solving Model (1), a drawback is the scalability of the approach and how quickly we can use it to derive dynamic solutions using a rolling-horizon computational framework. We will later show that the optimization models do not scale well, and they are not able to solve small- or medium-sized instances within two-hour computational
time limit. In this section, based on the spatial-temporal features of demand in NEMT-types of delivery services, we design vehicle routes and service schedules using machine learning and data classification algorithms. The goal is to improve computational time and to derive easy-to-implement decision policies under diverse sources of uncertainties. The main idea of these heuristics is to break the first-stage assignment-scheduling problem into two steps. In the first step, we cluster \(|I|\) customers into \(|J|\) groups based on their O-D pairs’ similarities using data clustering methods (Jain, 2010), such as K-means, K-medoids, etc. We can also modify and improve the initial solutions by ensuring that customers’ time windows do not significantly overlap in the same cluster. Then, we assign each cluster of customers to a vehicle and plan a schedule on each vehicle, which can be solved in parallel, based on sorted time windows of the customers in each corresponding cluster. (Note that once we know the customer-to-vehicle assignment and their service order, deciding an optimal schedule such as vehicle arrival time at each individual customer can be done quickly via solving a small-size linear programming model (3) described in Section 3.2.) We describe details of the two heuristics, K-means and K-means with swap, in Sections 4.1 and 4.2, respectively, for clustering geographically similar customers.

4.1 Heuristic 1: K-means

For each customer \(i\), we use Google API to extract the latitude and longitude of the origin and destination, denoted by \(O_{\text{lat}}^i, O_{\text{long}}^i, D_{\text{lat}}^i, D_{\text{long}}^i\). Then we get a point-by-feature matrix \(\{d_i\}_{i=1}^{|I|}\) where each \(d_i\) is a 4-dimensional real vector representing the geographical information of customer \(i\), i.e., \(d_i = (O_{\text{lat}}^i, O_{\text{long}}^i, D_{\text{lat}}^i, D_{\text{long}}^i)\). Via K-means clustering, we aim to partition the \(|I|\) data points into \(|J|\) \((\leq |I|)\) sets \(S = \{S_1, S_2, ..., S_{|J|}\}\) to minimize the within-cluster sum of squares (WCSS).

Formally, the problem can be cast as:

\[
\min_{m, \mu} \sum_{i \in I} \sum_{j \in J} m_{ji} ||d_i - \mu_j||^2 \tag{5}
\]

s.t. \(\sum_{j \in J} m_{ji} = 1, \forall i \in I,\)

\(m_{ji} \in \{0, 1\}, \forall j \in J, i \in I,\)

where \(\mu_j\) is the mean (centroid) of cluster \(S_j\) calculated by (7), and \(m_{ji} = 1\) if data point \(d_i\) belongs to cluster \(j\); and \(m_{ji} = 0\) otherwise.

The K-means is a special form of the well-known Expectation-Maximization (EM) algorithm (Moon, 1996), where in our case, the E-step is assigning the data points to the closest cluster and
the M-step is computing the centroid of each cluster. Specifically, we first randomly select \(|J|\) data points \(d_{i_1}, d_{i_2}, \ldots, d_{i_{|J|}}\) as the centroids where \(i_j \in I\) for all \(j \in J\). In the E-step, we fix \(\mu_j = d_{i_j}\) for all \(j \in J\) and solve problem (5), leading to the following optimal solution:

\[
m^*_ji = \begin{cases} 
1, & \text{if } j = \arg \min_{j' \in J} \|d_i - \mu_{j'}\|^2, \\
0, & \text{otherwise.}
\end{cases}
\] (6)

In the M-step, given the optimal assignment \(m_{ji} = m^*_ji\) for all \(i \in I, j \in J\), we optimize the objective function (5) over \(\mu\) to obtain an updated set of centroids:

\[
\mu^*_j = \frac{\sum_{i=1}^{|I|} m_{ji} d_i}{\sum_{i=1}^{|I|} m_{ji}}, \quad \forall j \in J.
\] (7)

We then fix \(\mu_j = \mu^*_j\) for all \(j \in J\) and keep iterating over these two steps until there is no change to the centroids, i.e., the assignment of data points to clusters is not changing. We summarize the detailed step of the K-means clustering algorithm in Algorithm 2.

**Algorithm 2** Use K-means to cluster \(|I|\) customers into \(|J|\) groups

1: Given O-D pairs of \(|I|\) customers, we use Google API to extract a point-by-feature matrix \(\{(O_{lat_i}, O_{long_i}, D_{lat_i}, D_{long_i})\}_{i=1}^{|I|}\).
2: Data Standardization: Re-scale the data matrix along each column to get mean 0 and standard deviation 1.
3: Initialization: randomly select \(|J|\) data points for the centroids without replacement.
4: while the assignment of data points to clusters is changing do
5: \hspace{1em} **Assignment step:** assign each data point to the cluster with the nearest centroid following Equation (6).
6: \hspace{1em} **Update step:** compute the centroids for the clusters by taking the average of the all data points that belong to each cluster following Equation (7).
7: end while

Given \(|J|\) clusters of customers, we assign one vehicle to each cluster and then solve Model (1) with fixed \((x, y)\)-values but without the second-stage cost to obtain a schedule and routing plan efficiently, which we denote as KM for short. One can also incorporate the second-stage cost as we discuss in the next heuristic.

### 4.2 Heuristic 2: K-means with swap

Algorithm 2 does not consider the information of time windows when performing clustering. Therefore, we may end up with a cluster of customers who have very similar planned start time and short time windows, resulting in either infeasible solutions or extremely long waiting time for some customers in the scheduling phase. Moreover, as we cannot control the number of data points in each cluster, some vehicles may have extremely high demand volumes while others are idle in most of the
operation time. In this heuristic, we propose a swapping method to distribute all customers more evenly to each vehicle, in terms of their time-window distributions and the number of customers in each cluster.

Specifically, we first apply Algorithm 2 to obtain an initial clustering result \( S = \{ S_1, S_2, ..., S_J \} \). Let \( S_j^k \) be the \( k^{th} \) element of cluster \( j \). For each cluster \( j \in J \), we evaluate the distance of time windows between any two adjacent customers \( S_j^k \) and \( S_j^{k+1} \). If the distance is smaller than a given threshold \( T \), then we re-assign either \( S_j^k \) or \( S_j^{k+1} \) to the cluster \( j_{\min} \) that has the smallest number of customers currently, i.e., \( j_{\min} = \arg \min_{j \in J} \text{length}(S_j) \), where we use \( \text{length}(S_j) \) to represent the number of customers in cluster \( S_j \). The selection criteria is to make sure that after inserting one of the customers into cluster \( j_{\min} \), the distances of time windows between the customer and the previous/next ones are no less than the threshold \( T \). As a result, we can ensure that there is enough time for each vehicle to transit between customers, and the numbers of customers in all the clusters are similar. We summarize the algorithmic details in Algorithm 3.

Similarly, after obtaining the assignment decisions from Algorithm 3, one can solve Model (1) either with the second-stage cost \( Q(u, r, z, \omega) \) to account for the randomness (denoted as KMSS), or without it to obtain solutions in a quick fashion (denoted as KMS). We will compare these two approaches in Section 5 later.

## 5 Computational Results

We compare different approaches for the static and dynamic vehicle routing and service scheduling problem using a diverse set of instances generated based on features of real data collected from operating Ford Motor Company’s NEMT service in 2019. We conduct in-sample and out-of-sample tests of Model (1) and the heuristic-based Algorithms 2 and 3 for handling uncertain service duration and cancellations. In Section 5.1, we describe detailed parameter settings in the baseline case and our experimental setup. We vary parameter choices to conduct sensitivity studies and report in-sample results in Section 5.2, and present out-of-sample results’ comparison between the optimization model and clustering-based heuristics in Section 5.3. In Section 5.4, we present the SAA analysis results, and in Sections 5.5 and 5.6, we present the results of applying the rolling horizon approach on small- and large-scale instances, respectively. We use Gurobi 9.0.3 coded in Python 3.6.8 for solving all mixed-integer programming models. Our numerical tests are conducted on a Windows 2012 Server with 128 GB RAM and an Intel 2.2 GHz processor.
Algorithm 3 K-means with swap to improve the clustering results by K-means in Algorithm 2.

1: Perform Algorithm 2 to get an initial clustering result $S = \{S_1, S_2, ..., S_\mid J\}$.
2: Set the iteration number $\ell = 1$ and the maximum iterations to $\ell_{\text{max}}$.
3: while not converged and $\ell < \ell_{\text{max}}$ do
   4: Calculate the cluster index with the smallest length, i.e., $j_{\text{min}} = \arg\min_{j \in J} \text{length}(S_j)$.
5: for $j = 1, \ldots, \mid J \mid$ and $j \neq j_{\text{min}}$ do
6: Sort customers in $S_j$ based on their time windows.
7: Set $k = 0$.
8: while $k < \text{length}(S_j) - 1$ do
9: Denote prev$_k$ and next$_k$ as the previous and next customer index of customer $S_k^j$ that belong to cluster $j_{\text{min}}$.
10: if $|\bar{a}_k^j - \bar{a}_{k+1}^j| < T$ then
11: if $|\bar{a}_{k+1}^j - \bar{a}_{\text{prev}_{k+1}}^j| \geq T$ and $|\bar{a}_{k+1}^j - \bar{a}_{\text{next}_{k+1}}^j| \geq T$ then
12: Re-assign customer $S_k^j$ to cluster $j_{\text{min}}$.
13: else if $|\bar{a}_k^j - \bar{a}_{\text{prev}_k}^j| \geq T$ and $|\bar{a}_k^j - \bar{a}_{\text{next}_k}^j| \geq T$ then
14: Re-assign customer $S_k^j$ to cluster $j_{\text{min}}$.
15: else
16: $k = k + 1$.
17: end if
18: else
19: $k = k + 1$.
20: end if
21: end while
22: end for
23: Check for convergence: if in every cluster, the time windows of any two adjacent customers are no less than the threshold $T$, then set converged to True; otherwise, set converged to False.
24: $\ell = \ell + 1$.
25: end while

5.1 Experimental design and setup

We generate customer’s time windows according to the temporal demand density reported by Ford GoRide Health team, shown in Figure 5 where the $x$-axis represents the requested pick-up time in hours and the $y$-axis indicates the kernel density. From Figure 5, vehicles start to operate at 4am and all the service end at 7pm, yielding a total operational time of 15 hours. Therefore, $L_j = 900$ minutes for all $j \in J$. For each customer $i \in I$, we sample the earliest pick-up time $a_i \in [0, 900]$ following the given density function, and then set $\bar{a}_i = a_i + 30$, meaning that each customer has a 30-minute time window.

During Year 2019, the NEMT service was operated in Sterling Heights, Wayne, Southfield, Dearborn, Taylor and Ann Arbor in Southeast Michigan, mainly for transporting patients who are elderly, disabled, or have chronic disease from and to their medical requests. We display the population estimate, percentage of people who are either over 65 years old or disabled in the six
cities in the first three columns of Table 2, based on the most updated information posted on U.S. Census Bureau (2010). In Table 2, we also calculate the number of target customers and the corresponding demand ratios in each city in the last two columns. In our baseline case, the total number of customers and vehicles in Southeast Michigan are \(|I| = 100, \ |J| = 20\), respectively, and we also decompose the whole service area into six service regions (cities) while distributing all the customers/vehicles to each city according to their demand ratios, with one depot in each city.

![Figure 5: Density of requested pick-up time during 4am to 7pm in a daily base.](image)

**Table 2:** Distributions of elderly (over 65 years old) and populations with disability based on census data in six cities in Southeast Michigan

<table>
<thead>
<tr>
<th>City</th>
<th>population</th>
<th>% of customers</th>
<th># of customers</th>
<th>proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sterling Heights</td>
<td>132,964</td>
<td>26%</td>
<td>34,571</td>
<td>29%</td>
</tr>
<tr>
<td>Wayne</td>
<td>16,896</td>
<td>28.5%</td>
<td>4,815</td>
<td>4%</td>
</tr>
<tr>
<td>Southfield</td>
<td>73,158</td>
<td>31.1%</td>
<td>22,752</td>
<td>19%</td>
</tr>
<tr>
<td>Dearborn</td>
<td>94,333</td>
<td>21.2%</td>
<td>19,999</td>
<td>17%</td>
</tr>
<tr>
<td>Taylor</td>
<td>61,148</td>
<td>29.4%</td>
<td>17,978</td>
<td>15%</td>
</tr>
<tr>
<td>Ann Arbor</td>
<td>121,890</td>
<td>15.6%</td>
<td>19,015</td>
<td>16%</td>
</tr>
</tbody>
</table>

We select representative hospitals and senior housing locations in the six cities and mark them in red and blue respectively in Figure 6, with a total of 23 hospitals and 48 senior housing locations, from which we can sample O-D pairs of all service requests received. Then we use Google API to calculate the average travel time between each pair of the sampled locations, serving as the empirical mean of the random travel time. For example, we use \(\tilde{\tau}_{O_i,D_i}\) to denote the empirical mean of the random travel time \(\tau_{O_i,D_i}\) from \(O_i\) to \(D_i\), which follows a normal distribution \(N(\tilde{\tau}_{O_i,D_i}, \tilde{\tau}_{O_i,D_i} \ast \sigma)\) with the standard deviation \(\sigma\) being 0.2 in the baseline case.
The service in the customers’ origins/destinations mainly includes loading/unloading them to/from the vehicle, which takes 20/5 minutes on average in the Ford’s NEMT system. Therefore, we let the service duration \( \hat{T}_{O_i} \) and \( \hat{T}_{D_i} \) follow normal distributions \( \mathcal{N}(20, 20 \cdot \sigma) \) and \( \mathcal{N}(5, 5 \cdot \sigma) \), respectively. Recall that we use \( q_i(\omega) \) to denote cancellation status of customer \( i \) in each scenario \( \omega \in \Omega \), which equals to 1 if the customer shows up and 0 otherwise. We sample all the \( q_i(\omega) \)-values following a Bernoulli distribution with showing-up probability = 0.89 as according to Ford, the cancellation rate of all trips is 11% in 2019. According to our discussions with Ford GoRide Health team, we set the daily operational cost of a vehicle as \( c_j = $240 \) for all vehicles \( j \in J \) and set per minute penalty cost of vehicle being idle, customer waiting, and overtime as \( c^l = $1 \), \( c^w = $2 \), \( c^o = $10 \), respectively. As Ford hires vans and drivers on a daily basis, they do not have significant cost associated with the total travel time and accordingly, we set the per minute penalty cost of vehicles’ travel time as \( c^d = $0 \), but will present the total travel time results for comparing different approaches.

We focus on the operations of NEMT in Ann Arbor, Michigan in Sections 5.2–5.5, which has \( |I| = 100 \times 16\% = 16 \) customers and \( |J| = 20 \times 16\% = 3 \) vehicles in the baseline case according to Table 2. For the in-sample tests of TS-MILP, the number of scenarios is set to \( |\Omega| = 10 \), and we evaluate solutions given by TS-MILP and clustering-based heuristics on 1000 independently generated out-of-sample scenarios. Note that in the objective function (3a), we do not penalize the first idle time of each vehicle (i.e., \( l^i_0 \)), as we can always inform the vehicles to start at the requested pick-up time of their first customer. Therefore, we exclude \( l^i_0 \) in our results when calculating average idle time per vehicle. We compute instances based on the whole Southeast Michigan service zone in Section 5.6 for demonstrating the scalability results of TS-MILP and service clustering heuristics.

### 5.2 In-sample results and sensitivity analysis of TS-MILP

Using the baseline setting, we first vary the standard deviation \( \sigma \) from 0.2 to 0.8 to see the effects of the variances of travel time and service duration on in-sample solutions, reported in Table 3. We then vary the showing-up probability for each customer from 0.2 to 0.8 while keeping \( \sigma = 0.2 \) to illustrate the impacts of service cancellation, presented in Table 4.

In Tables 3 and 4, ID, OT, TT and WT denote the average idle/overtime/total travel time (in minutes) per vehicle per scenario and average waiting time (in minutes) per customer per scenario across all in-sample scenarios and the last two columns display the overall objective value of Model (1) in dollars and the computational time in seconds, respectively. From Tables 3 and 4, when
Figure 6: A map of Southeast Michigan with 48 senior housing locations marked by blue and 23 hospitals marked by red.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>ID (min.)</th>
<th>WT (min.)</th>
<th>OT (min.)</th>
<th>TT (min.)</th>
<th>Obj. ($)</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>218.08</td>
<td>0.41</td>
<td>0.00</td>
<td>217.89</td>
<td>1387.34</td>
<td>103.00</td>
</tr>
<tr>
<td>0.4</td>
<td>218.80</td>
<td>1.13</td>
<td>0.00</td>
<td>222.28</td>
<td>1412.71</td>
<td>138.71</td>
</tr>
<tr>
<td>0.6</td>
<td>222.68</td>
<td>1.96</td>
<td>0.00</td>
<td>221.69</td>
<td>1450.77</td>
<td>186.22</td>
</tr>
<tr>
<td>0.8</td>
<td>229.40</td>
<td>2.69</td>
<td>0.09</td>
<td>220.16</td>
<td>1496.81</td>
<td>221.67</td>
</tr>
</tbody>
</table>

Table 3: In-sample results of TS-MILP with varying $\sigma$

Table 4: In-sample results of TS-MILP with varying show-up probability $\hat{q}_i$

<table>
<thead>
<tr>
<th>$\hat{q}_i$</th>
<th>ID (min.)</th>
<th>WT (min.)</th>
<th>OT (min.)</th>
<th>TT (min.)</th>
<th>Obj. ($)</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>325.04</td>
<td>2.21</td>
<td>0.00</td>
<td>71.50</td>
<td>1765.91</td>
<td>95.15</td>
</tr>
<tr>
<td>0.4</td>
<td>311.35</td>
<td>1.02</td>
<td>0.00</td>
<td>116.87</td>
<td>1686.78</td>
<td>111.71</td>
</tr>
<tr>
<td>0.6</td>
<td>276.51</td>
<td>0.50</td>
<td>0.00</td>
<td>158.02</td>
<td>1565.64</td>
<td>134.08</td>
</tr>
<tr>
<td>0.8</td>
<td>237.51</td>
<td>0.35</td>
<td>0.00</td>
<td>198.84</td>
<td>1443.62</td>
<td>132.05</td>
</tr>
</tbody>
</table>

increasing the variance of travel time and service duration, the overall objective values and the average waiting time both increase; when fewer customers cancel their reservations, the overall objective values, average idle time and waiting time are better. Therefore, to obtain satisfactory quality-of-service, one key component is to maintain low variance of travel time and service duration and low cancellation rates.

To illustrate how the vehicles’ routes and schedules change in response to different customers’ show-up chances, we present each vehicle’s operational status when $\hat{q}_i = 0.2, 0.8$ in Figure 7, where
the $x$-axis denotes the time during operation with 4am being 0 and 7pm being 900 minutes, and the $y$-axis denotes the three status of vehicles: working, waiting and idle. From Figures 7(a)–(c), when customers all show up with a lower probability, the three vehicles start to work at different time of day (i.e., around 7am, 12pm, 5pm, respectively), so as to minimize the average idle time. On the contrary, when $\hat{q}_i = 0.8$, the tasks are distributed to vehicles more evenly, as depicted in Figures 7(d)–(f).

![Figures 7(a)–(c)](image1)
![Figures 7(d)–(f)](image2)

Figure 7: Illustration of vehicle operational status with different $\hat{q}_i$.

Next we fix the baseline setting and vary $|J|$ from 3 to 5 and $|I|$ from 16 to 32 in Table 5, where we mark the optimality gaps of the instances that cannot be solved within 7200 seconds in the bracket. From Table 5, the waiting time per customer is almost negligible compared to the

| $|J|$ | $|I|$ | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($) | Time (sec.) |
|-----|-----|----------|----------|----------|----------|----------|-------------|
| 3   | 16  | 218.08   | 0.41     | 0.00     | 217.89   | 1387.34  | 103.00      |
| 4   | 32  | 235.30   | 0.58     | 0.00     | 168.95   | 1919.62  | 105.56      |
| 5   | 16  | 188.30   | 0.57     | 0.00     | 132.70   | 2159.62  | 722.88      |
| 3   | 32  | 218.13   | 0.92     | 0.00     | 417.87   | 1433.20  | 7200.00     |
| 4   | 32  | 246.30   | 1.21     | 0.00     | 321.70   | 2022.40  | 7200.00     |
| 5   | 32  | 220.80   | 1.75     | 0.00     | 257.70   | 2416.20  | 7200.00     |

23
idle time per vehicle. Moreover, the total travel time per vehicle decreases when we have more vehicles and it almost gets doubled when we increase the number of customers from 16 to 32. It is also noteworthy that TS-MILP cannot be solved to optimality within 2 hours when we have 32 customers, which brings the need to use heuristics to derive sub-optimal solutions in a quick fashion.

5.3 Out-of-sample tests and results of different approaches

We first compare the out-of-sample results of solving Model (1) using off-the-shelf solvers directly and using the classical Benders’ decomposition algorithm presented in Appendix B (Benders, 1962) in Table 6, where we record the relative gaps between the upper bound (UB) and lower bound (LB) provided by Benders’ decomposition and the gaps of its upper bound and the optimal objective value (OPT) in the last two columns, respectively. We terminate Benders’ decomposition algorithm in 25 and 50 iterations, and denote them by Benders-25 and Benders-50, correspondingly. From Table 6, Benders’ decomposition fails to solve the problem to optimality within 24 hours (or 50 iterations), which performs much worse than directly solving it using a state-of-the-art solver. This observation was also revealed in Zhan et al. (2021), where the authors pointed out that “the lower and upper bounds are improved quite slowly (in our experiments, the lower bounds hardly increase within several hours, even after hundreds of iterations).” Because of that, we will compare the results of solving Model (1) using Gurobi with heuristic approaches in the remaining of the paper.

Table 6: Out-of-sample results of solving TS-MILP via Gurobi and Benders’ decomposition with $|J| = 3$ and $|I| = 16$

| $|J|/|I|$ | Method | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($) | Time (sec.) | (UB - LB)/LB | (UB - OPT)/OPT |
|--------|--------|---------|----------|---------|---------|---------|-----------|-------------|-------------|
| 3 16   | TS-MILP | 221.04  | 0.53     | 0.02    | 215.23  | 1400.40 | 103.00    | N.A.        | N.A.        |
|       | Benders-25 | 347.60  | 8.04     | 0.00    | 219.24  | 2019.91 | 8152.35   | 90.49%      | 45.07%      |
|       | Benders-50 | 220.60  | 0.72     | 0.00    | 226.08  | 1404.94 | 89606.44  | 5.78%       | 1.25%       |

Before proceeding to other heuristics, we examine the out-of-sample performance of solving Model (1) with different in-sample scenarios $|\Omega|$. Table 7 presents the results where we vary $|\Omega|$ from 10 to 100, and the last column indicates the computational time for the in-sample tests. From Table 7, there are no significant result improvements when we increase the sample size, while the computational time increases drastically. As a result, we continue using $|\Omega| = 10$ in our subsequent tests.

Table 7: Out-of-sample results of solving TS-MILP via Gurobi for various in-sample scenarios $|\Omega|

In Table 8, we compare the out-of-sample results between TS-MILP in Section 3 and the three
heuristics proposed in Section 4, namely K-means, K-means with swap and K-means with swap and
the second-stage cost (KM, KMS and KMSS for short). We also set the maximum number of
swapping steps $\ell_{\text{max}} = 5$ and the threshold $T = 50$ at default in Algorithm 3. We vary the value of $T$ later in Table 9.

### Table 7: Out-of-sample results of TS-MILP with varying in-sample scenario size $|\Omega|$  

| $|J|$ | $|I|$ | $|\Omega|$ | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($\$)$ | Time (sec.) |
|-----|-----|-----|---------|---------|---------|---------|---------|--------|---------|
| 10  | 16  | 50  | 221.04  | 0.53    | 0.02    | 215.23  | 1400.40 | 103.00 |
| 100 | 16  | 50  | 221.52  | 0.40    | 0.00    | 220.14  | 1397.53 | 816.26 |

### Table 8: Out-of-sample tests and results of different approaches with varying $|J|$ and $|I|$  

| $|J|$ | $|I|$ | Method | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($\$)$ | Time (sec.) |
|-----|-----|--------|---------|---------|---------|---------|---------|--------|
| 3   | 16  | TS-MILP| 221.04  | 0.53    | 0.02    | 215.23  | 1400.40 | 103.00 |
|     |     | KM     | 435.40  | 2.22    | 0.24    | 203.52  | 2104.54 | 0.13   |
|     |     | KMS    | 434.65  | 0.12    | 0.00    | 207.14  | 2027.80 | 0.13   |
|     |     | KMSS   | 404.92  | 0.10    | 0.00    | 207.14  | 1937.87 | 5.19   |
|     |     | TS-MILP| 237.34  | 0.68    | 0.04    | 167.19  | 1932.62 | 105.56 |
|     |     | KM     | 425.00  | 0.59    | 0.00    | 153.24  | 2678.99 | 0.15   |
|     |     | KMS    | 420.29  | 0.12    | 0.00    | 152.01  | 2644.99 | 0.18   |
|     |     | KMSS   | 390.44  | 0.11    | 0.00    | 152.00  | 2525.36 | 6.66   |
|     |     | TS-MILP| 189.93  | 0.68    | 0.03    | 131.06  | 2172.79 | 722.88 |
|     |     | KM     | 441.20  | 0.09    | 0.00    | 123.15  | 3408.86 | 0.22   |
|     |     | KMS    | 446.32  | 0.10    | 0.00    | 122.74  | 3434.83 | 0.21   |
|     |     | KMSS   | 411.29  | 0.09    | 0.00    | 123.16  | 3259.55 | 8.16   |
|     |     | TS-MILP| 218.07  | 1.13    | 0.00    | 416.53  | 1446.53 | 7200.00 |
|     |     | KM     | 331.07  | 25.48   | 8.47    | 408.24  | 3598.21 | 0.95   |
|     |     | KMS    | 327.74  | 14.02   | 1.55    | 410.15  | 2646.75 | 0.37   |
|     |     | KMSS   | 297.54  | 11.55   | 1.07    | 409.87  | 2383.79 | 33.96  |
|     |     | TS-MILP| 243.93  | 1.79    | 0.00    | 320.98  | 2050.40 | 7200.00 |
|     |     | KM     | 396.71  | 7.38    | 0.00    | 309.88  | 3019.21 | 0.81   |
|     |     | KMS    | 416.40  | 2.74    | 0.00    | 312.52  | 2800.92 | 0.38   |
|     |     | KMSS   | 386.27  | 0.92    | 0.00    | 312.52  | 2563.91 | 38.02  |
|     |     | TS-MILP| 220.63  | 2.15    | 0.00    | 258.56  | 2440.83 | 7200.00 |
|     |     | KM     | 387.86  | 4.05    | 0.00    | 340.26  | 3398.61 | 0.59   |
|     |     | KMS    | 457.65  | 1.44    | 0.00    | 245.49  | 3580.42 | 0.35   |
|     |     | KMSS   | 421.92  | 0.81    | 0.00    | 244.91  | 3361.18 | 39.85  |
From Table 8, TS-MILP always obtains the best out-of-sample performance in terms of the idle time and overall objective values, as it is designed to optimize the expected objectives under uncertainty. All the heuristics perform significantly worse in idle time while slightly improve the waiting time and total travel time. On the other hand, they reduce the computational time from hundreds of seconds to less than ten seconds for the small-scale instances when \(|I| = 16\) based on Ann Arbor. When \(|I| = 32\), the heuristics can still solve the problems within 40 seconds, while TS-MILP cannot be optimized within 2 hours. Moreover, KMS improves the overall objective values of KM, which is further reduced by KMSS while maintaining the computational efficiency.

Table 9: Out-of-sample tests and results of K-means with swap with varying threshold \(T\)

| \(|J|\) | \(|I|\) | \(T\) (min.) | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. (\$) | Time (sec.) |
|-------|-------|--------------|-----------|-----------|-----------|-----------|------------|------------|
| 3     | 16    | 30           | 435.40    | 2.22      | 0.24      | 203.52    | 2104.54    | 0.13       |
|       |       | 40           | 434.80    | 0.12      | 0.00      | 207.62    | 2028.24    | 0.16       |
|       |       | 50           | 434.65    | 0.12      | 0.00      | 207.14    | 2027.80    | 0.13       |

As can be seen from Algorithm 3, the threshold \(T\) plays an important role in determining the swap between customers and the termination of the algorithm. In Table 9, we fix \(|J| = 3\), \(|I| = 16\) and test K-means with swap where we vary the threshold \(T\) from 30 to 50 minutes. With more transit time intentionally left for adjacent customers, all performance metrics get improved, where the improvements from \(T = 30\) to \(40\) are much more significant than the ones from \(40\) to \(50\). As a result, in the following tests, we continue to fix \(T = 50\). We also present the results of K-means-based heuristics with different swapping steps \(\ell_{\text{max}}\) and different input feature matrix \(\{d_{i}\}_{i=1}^{\mid I\mid}\) in Appendix C.

5.4 Comparison between stochastic and deterministic approaches

We compare the in-sample and out-of-sample performance of TS-MILP and its deterministic counterpart, where we generate \(|\Omega|\) in-sample scenarios to obtain an optimal TS-MILP solution and use the empirical mean of these \(|\Omega|\) in-sample scenarios to obtain a deterministic optimal solution. Then we evaluate these two solutions on the same 1000 out-of-sample scenarios based on the overall objective values.

We generate 10 independent sets of scenario samples, each of size \(|\Omega|\), and conduct the tests independently for each set of scenarios. We calculate the average objective values of in-sample and out-of-sample tests in Table 10, and display the box plot in Figure 8.
Table 10: Average objective values of in-sample and out-of-sample tests and their gaps (across 10 replications) using TS-MILP and its deterministic counterpart.

<table>
<thead>
<tr>
<th>TS-MILP IS</th>
<th>TS-MILP OS</th>
<th>Gap 1</th>
<th>DT IS</th>
<th>DT OS</th>
<th>Gap 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1408.21</td>
<td>1420.11</td>
<td>0.85%</td>
<td>1337.96</td>
<td>1442.07</td>
</tr>
</tbody>
</table>

In both Table 10 and Figure 8, “TS-MILP IS”, “TS-MILP OS” and “Gap 1” denote the average objective values of in-sample, out-of-sample tests and the gap between the two, while “DT IS”, “DT OS” and “Gap 2” represent the ones of the deterministic counterpart, respectively.

From Table 10 and Figure 8, the gap between the average in-sample and out-of-sample objective values in TS-MILP is 0.85%, indicating that the optimal solutions computed by $|\Omega|$ in-sample scenarios also perform well in the 1000 out-of-sample performance. However, in the deterministic counterpart, the gap between the average out-of-sample and in-sample objective values is 7.78%. Although solving the deterministic model with empirical means can return a better in-sample result, the out-of-sample performance could be much worse compared to the one of TS-MILP.

5.5 Results of rolling horizon method on Ann Arbor instances

In this section, we present results from the rolling horizon approach in Section 3.3 for optimizing real-time vehicle routing and service scheduling. In Algorithm 1, first, Model (1) is solved to obtain an initial vehicle-customer assignment. Then, in each period, demand is realized and fulfilled
on a first-come, first-served basis, and we adaptively change the assignment and schedule plan. Alternatively, one can also use the two heuristics to match vehicle-customer pairs in each period, which gives us a combination of Algorithm 1 with Algorithms 2 and 3. We still focus on operations in Ann Arbor, which has 16 initial customers with reservations one-day ahead, and then we set another 8 real-time customers who request at least 30 minutes in advance, and their time windows are also drawn from the density function presented in Figure 5. Therefore, we have a total of $|J| = 16 + 8 = 24$ customers and in Table 11, we present the results of the rolling horizon method combined with different algorithms, where the last column displays the average computational time across all periods.

| $|J|$ | $|I|$ | Method | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($\$$) | Time (sec.) |
|-----|-----|--------|------------|-----------|-----------|-----------|-------------|-------------|
| 3   | 24  | TS-MILP | 179.67     | 2.38      | 0.00      | 487.33    | 1373.24    | 55.23       |
|     |     | KM     | 217.33     | 4.71      | 0.00      | 480.00    | 1598.00    | 0.16        |
|     |     | KMS    | 224.00     | 6.08      | 0.00      | 473.33    | 1684.00    | 0.15        |
|     |     | KMSS   | 240.00     | 0.33      | 0.00      | 420.33    | 1456.00    | 3.05        |
| 4   | 24  | TS-MILP | 250.75     | 0.25      | 0.00      | 383.50    | 1975.00    | 45.05       |
|     |     | KM     | 220.25     | 3.63      | 0.00      | 450.50    | 2015.00    | 0.14        |
|     |     | KMS    | 262.75     | 0.67      | 0.00      | 408.00    | 2043.00    | 0.11        |
|     |     | KMSS   | 259.75     | 0.54      | 0.00      | 379.25    | 2025.00    | 2.63        |
| 5   | 24  | TS-MILP | 214.00     | 1.92      | 0.00      | 473.40    | 2362.00    | 235.41      |
|     |     | KM     | 243.80     | 1.33      | 0.00      | 404.00    | 2483.00    | 0.14        |
|     |     | KMS    | 247.40     | 0.38      | 0.00      | 400.40    | 2455.00    | 0.16        |
|     |     | KMSS   | 258.20     | 0.92      | 0.00      | 353.60    | 2535.00    | 3.86        |

From Table 11, when we increase the number of vehicles $|J|$ from 3 to 4, the average waiting time decreases while the average idle time increases. The differences between TS-MILP and the heuristic approaches in terms of the overall objective values are much smaller than the ones in the static setting, and the heuristics even improve the waiting/travel time in some instances. Moreover, the heuristic approaches maintain the computational efficiency by solving all the instances within 4 seconds, which sheds light on the applicability of these heuristics in dynamic settings.

Next we present the performance of the rolling horizon method combined with TS-MILP in Table 12, where we vary the sample size $|\Omega|$ from 10 to 100. From Table 12, with more in-sample scenarios, although the average waiting and travel time get reduced, vehicles are idle for a longer period of time and the overall objective cost also increases. This is because we do not take into
to account the uncertainty of future customers when making decisions, and enlarging the sample size for the current stage’s uncertainty would not necessarily help in the dynamic environment.

Table 12: Results of rolling horizon algorithm solved by TS-MILP with varying in-sample scenario size $|\Omega|$.

| $|J|$ | $|I|$ | $|\Omega|$ | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. | Time (sec.) |
|------|------|----------|----------|----------|----------|----------|------|-------------|
| 10   | 10   | 10       | 179.67   | 2.38     | 0.00     | 487.33   | 1373.24 | 55.23       |
| 3    | 24   | 50       | 221.67   | 1.21     | 0.00     | 443.00   | 1443.08 | 92.15       |
| 100  | 100  | 100      | 263.33   | 0.50     | 0.00     | 403.33   | 1534.00 | 162.32      |

5.6 Results of large-scale operations using clustering-based heuristics

Having witnessed the efficiency of clustering-based heuristics, we now present the performance of them on large instances based on Southeast Michigan with six operating cities, and vary the number of vehicles from 20 to 40 and the number of customers from 100 to 300 in Table 13. In these instances, we set the maximum number of swapping steps $\ell_{\text{max}} = 5$ in Algorithm 3 for implementing KMS and KMSS.

From Table 13, when increasing the number of customers, the idle time per vehicle decreases while the waiting time per customer and the overtime per vehicle increase drastically. The waiting time and overtime also drop significantly with doubled vehicles. Moreover, KMSS improves the performance drastically by shortening waiting time per customer and overtime per vehicle.

Next, we present the results of using the rolling horizon method with KM, KMS and KMSS for operating Southeast Michigan in Table 14 and set the number of dynamically arrived customers as half of the number of customers with reservations, such that the total $|I|$ ranges from 150 to 450. Comparing Tables 14 with 13, the idle time per vehicle decreases, while the travel time per vehicle almost gets doubled as we include dynamically arriving customers. The computational time also decreases as we average among all the periods while later periods having much smaller sizes can be solved relatively quickly.

6 Conclusion

In this paper, we modeled a TS-MILP for solving static and dynamic vehicle routing and scheduling problem, where we applied rolling horizon method to solve the dynamic variant. To speed up computation, we proposed K-means-based heuristics to cluster geographically similar customers.
and then separately decide routing and scheduling plan in each cluster. We conducted various experiments based on data collected by Ford Motor Company’s GoRide Health team. Results indicate that the clustering-based heuristics can solve large-scale instances efficiently and effectively.

For future research, one possibility is to design a branch-and-price algorithm for solving the TS-MILP that, unlike many VRP variants, involves non-negligible service duration at each customer location and also multiple parameter uncertainties. The development of appropriate pricing subproblems to generate route-and-schedule-combined columns is challenging. Moreover, as the pick-up/drop-off locations are typically hospitals and senior apartments, a large portion of customers may share similar routes or the same origins/destinations. It would be beneficial to pool these customers together, which may reduce the total operational cost but increase individuals’ waiting time.

Table 13: Performance of clustering-based heuristics on large instances of Southeast Michigan

| $|J|$ | $|I|$ | Method | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($) | Time (sec.) |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 100 | KM | 335.38 | 7.43 | 0.69 | 173.98 | 13129.87 | 0.30 |
| | | KMS | 412.41 | 0.30 | 0.38 | 174.73 | 13185.72 | 0.22 |
| | | KMSS | 384.06 | 0.19 | 0.38 | 174.73 | 12595.56 | 12.35 |
| | 200 | KM | 286.14 | 42.32 | 31.58 | 336.90 | 33764.87 | 0.93 |
| | | KMS | 284.15 | 12.58 | 14.13 | 350.38 | 18343.01 | 0.55 |
| | | KMSS | 257.90 | 7.33 | 11.47 | 350.46 | 15184.04 | 85.86 |
| | 300 | KM | 207.27 | 126.76 | 137.80 | 517.21 | 112561.06 | 2.42 |
| | | KMS | 168.20 | 85.28 | 88.44 | 535.63 | 77017.12 | 1.29 |
| | | KMSS | 148.42 | 69.61 | 85.04 | 534.65 | 66542.21 | 335.05 |
| 40 | 100 | KM | 375.08 | 2.54 | 0.21 | 89.76 | 25194.28 | 0.34 |
| | | KMS | 411.96 | 0.04 | 0.19 | 90.24 | 26163.24 | 0.28 |
| | | KMSS | 382.86 | 0.02 | 0.19 | 90.24 | 24995.25 | 14.72 |
| | 200 | KM | 361.19 | 13.64 | 8.71 | 171.42 | 32986.82 | 1.12 |
| | | KMS | 417.96 | 0.89 | 4.58 | 172.40 | 28508.52 | 0.72 |
| | | KMSS | 388.05 | 0.39 | 4.58 | 172.40 | 27111.77 | 129.63 |
| | 300 | KM | 341.93 | 42.30 | 26.39 | 251.73 | 59211.74 | 2.20 |
| | | KMS | 365.13 | 3.17 | 5.16 | 253.41 | 28168.64 | 1.15 |
| | | KMSS | 335.96 | 0.98 | 4.44 | 253.41 | 25404.40 | 293.83 |
Table 14: Results of the rolling horizon approach with clustering-based heuristics on large-scale instances

| $|J|$ | $|I|$ | Method | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. ($) | Time (sec.) |
|-----|-----|-------|---------|----------|----------|----------|----------|-----------|-------------|
| 20  | 150 | KM    | 204.35  | 8.71     | 4.00     | 425.30   | 12301.00 | 0.15      |
|     |     | KMS   | 211.65  | 2.47     | 3.95     | 448.45   | 10563.00 | 0.13      |
|     |     | KMSS  | 213.80  | 1.16     | 0.15     | 414.70   | 9454.00  | 5.37      |
| 20  | 300 | KM    | 158.15  | 17.38    | 17.20    | 551.50   | 21833.00 | 0.33      |
|     |     | KMS   | 163.80  | 14.41    | 15.20    | 556.40   | 19762.00 | 0.31      |
|     |     | KMSS  | 129.90  | 6.68     | 9.45     | 557.40   | 13296.00 | 38.40     |
| 20  | 450 | KM    | 112.05  | 27.82    | 25.55    | 612.70   | 37191.00 | 0.69      |
|     |     | KMS   | 121.40  | 25.62    | 20.80    | 606.80   | 34442.00 | 0.72      |
|     |     | KMSS  | 87.45   | 14.90    | 34.40    | 641.10   | 26839.00 | 135.94    |
| 40  | 150 | KM    | 197.98  | 1.95     | 1.50     | 474.48   | 18703.00 | 0.26      |
|     |     | KMS   | 200.68  | 0.52     | 1.63     | 447.88   | 18433.00 | 0.23      |
|     |     | KMSS  | 210.08  | 0.13     | 0.08     | 385.18   | 18073.00 | 8.17      |
| 40  | 300 | KM    | 184.33  | 6.36     | 7.00     | 480.63   | 23589.00 | 0.61      |
|     |     | KMS   | 202.33  | 1.69     | 7.70     | 455.58   | 21789.00 | 0.51      |
|     |     | KMSS  | 205.98  | 0.51     | 3.45     | 409.10   | 19525.00 | 54.16     |
| 40  | 450 | KM    | 168.75  | 9.96     | 7.65     | 491.45   | 28378.00 | 1.24      |
|     |     | KMS   | 189.33  | 2.30     | 9.28     | 485.08   | 22955.00 | 0.88      |
|     |     | KMSS  | 169.30  | 1.71     | 4.45     | 472.93   | 19688.00 | 134.12    |

Acknowledgments

The research has been supported by the University of Michigan and Ford Motor Company Alliance program (grant number 000356-UM0222-H). Shen is supported in part by the NSF under grant #CMMI-1727618. The authors sincerely thank Yuhai Hu and Heng Liu at Ford Motor Company for helpful discussions.

References


Online Supplement of the Paper

"Integrated Vehicle Routing and Service Scheduling under Time and Cancellation Uncertainties with Application in Non-Emergency Medical Transportation"

A Extensions of Model (1)

A.1 Model Extension I: Allowing ride-pooling

Because there are usually multiple seats in a Ford van, it opens up the opportunity to pool multiple passengers together. Next, we discuss how to extend Model (1) to deal with multiple requests in a slot. Denote $h \in \mathbb{Z}_+$ as the pooling capacity. In the first stage, we need to replace Constraints (1e) and (1f) with the following:

\begin{align}
& u_j^{i, k} \geq 1 - \sum_{i \in I} u_j^{i, k}, \forall k \in K, j \in J \quad (8a) \\
& \sum_{i \in I} u_j^{i, k} \leq (1 - u_{\text{depot}, k}^j)h, \forall k \in K, j \in J \quad (8b) \\
& \sum_{i \in I} u_j^{i, k} \leq hx_j, \forall k \in K, j \in J \quad (8c) \\
& \sum_{i \in I} u_j^{i,k+1} \leq (\sum_{i \in I} u_j^{i,k})h, \forall k = 1, \ldots, |K| - 1, j \in J. \quad (8d)
\end{align}

where Constraints (8a) ensure that if there are no requests assigned to a service slot of a vehicle, then this vehicle will return to the depot (i.e., $u_{\text{depot}, k}^j = 1$); otherwise, Constraints (8b) prohibit $u_{\text{depot}, k}^j = 1$. Constraints (8c) enforce vehicles’ capacity for each service slot, and constraints (8d) prohibit assigning a request to a slot on a vehicle if an earlier slot is vacant.

Assuming that $q_i(\omega) = 1$, $\forall i \in I$, $\omega \in \Omega$ and considering the special case where $h = 2$, we present a revised two-stage model as follows.

\begin{align}
& \min \sum_{j \in J} \left( \sum_{k \in K} (c^w w_k^j(\omega) + c^l l_k^j(\omega)) + c^W W_j(\omega) + c^d V_j(\omega) \right) \quad (9a) \\
& \text{s.t. } w_1^j(\omega) - l_0^j(\omega) = \sum_{i_1, i_2 \in I} t_{\text{depot}, i_1 i_2}(\omega) u_{i_1, 1}^j u_{i_2, 1}^j - r_1^j, \forall j \in J, \quad (9b) \\
& w_{k+1}^j(\omega) - l_k^j(\omega) - w_k^j(\omega) = \left( r_k^j + \sum_{i_1, i_2 \in I} T_{i_1, i_2}(\omega) u_{i_1, k}^j u_{i_2, k}^j \right) \quad (9c)
\end{align}
where we omit the scenario index $\omega$ for notation simplicity and present the definitions of $t_{\text{depot},i_1i_2}$, $T_{i_1,i_2}$ and $t_{i_1i_2,i_3i_4}$, respectively below:

$$
t_{\text{depot},i_1i_2} = \begin{cases} 
t_{\text{depot},i_1}, & \text{if } i_1 = i_2 \\
\tilde{t}_{\text{depot},i_1i_2} - t_{\text{depot},i_1} - t_{\text{depot},i_2}, & \text{otherwise}
\end{cases}
$$

$$
T_{i_1,i_2} = \begin{cases} 
T_{i_1}, & \text{if } i_1 = i_2 \\
\tilde{T}_{i_1,i_2} - T_{i_1} - T_{i_2}, & \text{otherwise}
\end{cases}
$$

$$
t_{i_1i_2,i_3i_4} = \begin{cases} 
t_{i_1,i_3}, & \text{if } i_1 = i_2, \ i_3 = i_4 \\
\tilde{t}_{i_1,i_3i_4} - t_{i_1,i_3} - t_{i_1,i_4}, & \text{if } i_1 = i_2, \ i_3 \neq i_4 \\
\tilde{t}_{i_1,i_2,i_3} - t_{i_1,i_3} - t_{i_2,i_3}, & \text{if } i_1 \neq i_2, \ i_3 = i_4 \\
\tilde{t}_{i_1i_2,i_3i_4} - (t_{i_1,i_3i_4} + t_{i_2,i_3i_4} + t_{i_1i_2,i_3} + t_{i_1i_2,i_4}) + t_{i_1,i_3} + t_{i_2,i_3} + t_{i_1,i_4} + t_{i_2,i_4}, & \text{otherwise}
\end{cases}
$$

We illustrate the idea using $t_{\text{depot},i_1i_2}$ as an example and the rest follows the same logic. If only one customer $i_1$ is assigned to the first slot, then $t_{\text{depot},i_1i_2} = t_{\text{depot},i_1}$; otherwise, we assume customers $i_1$ and $i_2$ ($i_1 \neq i_2$) are assigned to the first slot, and the first travel time should be $\tilde{t}_{\text{depot},i_1i_2}$ representing the shortest travel time from the depot to either $i_1$ or $i_2$ depending on which customers’ origin is closer to the depot. However, $\sum_{i_1,i_2 \in I} t_{\text{depot},i_1i_2}(\omega)u_{i_1,1}^{j}u_{i_2,1}^{j} = t_{\text{depot},i_1} + t_{\text{depot},i_2} + t_{\text{depot},i_1i_2}$ which also include the first two extra terms. As a result, we set $t_{\text{depot},i_1i_2} = \tilde{t}_{\text{depot},i_1i_2} - t_{\text{depot},i_1} - t_{\text{depot},i_2}$ to cancel out the first two terms. Similarly, we define $\tilde{T}_{i_1,i_2}$ as the total service duration of customers $i_1$ and $i_2$, depending on the shortest path to pick up and drop off them, and $\tilde{t}_{i_1,i_3i_4}$, $\tilde{t}_{i_1i_2,i_3}$, $\tilde{t}_{i_1i_2,i_3i_4}$ as the shortest travel time from customer $i_1$ to the group of customers $i_3$, $i_4$, the group of customers $i_1$, $i_2$ to customer $i_3$, and the group of customers $i_1$, $i_2$ to the group of customers $i_3$, $i_4$, respectively.
A.2 Model Extension II: Enforcing deadlines for drop-off time

In the context of NEMT, it is important to deliver patients to their medical appointments on time. We denote \( b_i \geq \bar{a}_i \) as the requested drop-off time of customer \( i \) for all \( i \in I \) and aim to deliver each customer before this deadline. As the actual arrival time depends on the realized travel time and service duration, we cannot pose it as a hard constraint in the first stage. However, we can add a delayed drop-off time in the second-stage problem and minimize the expected penalty cost of it. Mathematically, we define \( d^j_k(\omega) \) as the delayed drop-off time for the \( k \)th slot on vehicle \( j \) in scenario \( \omega \) for all \( k \in K, j \in J, \omega \in \Omega \), which can be calculated by

\[
d^j_k(\omega) = \max \left\{ 0, r^j_k + w^j_k(\omega) + \sum_{i \in I} q_i(\omega)T_i(\omega)u^j_{i,k} - \sum_{i \in I} q_i(\omega)b_iu^j_{i,k} - (1 - \sum_{i \in I} q_i(\omega)u^j_{i,k})(r^j_k + w^j_k(\omega)) \right\},
\]

given a first-stage VRP solution \( u, r \) and scenario \( \omega \). Then we can add the following constraints into Model (3)

\[
d^j_k(\omega) \geq r^j_k + w^j_k(\omega) + \sum_{i \in I} q_i(\omega)T_i(\omega)u^j_{i,k} - \sum_{i \in I} q_i(\omega)b_iu^j_{i,k} - (1 - \sum_{i \in I} q_i(\omega)u^j_{i,k})(r^j_k + w^j_k(\omega)),
\forall k = 1, \ldots, |K| - 1, j \in J, (10)
\]

\[
d^j_k(\omega) \geq 0, \forall k = 1, \ldots, |K| - 1, j \in J.
\]

Note that there are bilinear terms \( u^j_{i,k}r^j_k \) and \( u^j_{i,k}w^j_k(\omega) \) on the right-hand-side of (10). As \( u^j_{i,k} \) are binary variables, we can provide exact reformulations using McCormick envelopes similar to (4a)–(4c).

A.3 Model Extension III: Allowing multiple depots

Model (1) considers one depot for all vehicles. In practice, vehicles may start and end at different depots. In this case, we specify a set of depots \( P \) and denote depot \( j \in P \) as the depot associated with vehicle \( j \) for all \( j \in J \). Then, we need to change all the variables related to vehicle \( j \)’s depot by depot \( j \) for all \( j \in J \). For example, \( u^j_{\text{depot},k} \) should be changed to \( u^j_{\text{depot},j,k} \) and Constraints (3b) become \( w^j_1(\omega) - l^j_0(\omega) = \sum_{i \in I} t_{\text{depot},i}(\omega)u^j_{i,1} - r^j_1, \forall j \in J \).

B Benders’ Decomposition

To solve Model (1), a classical way is to apply Benders’ decomposition (Benders, 1962), which we introduce as follows. For notation simplicity, we denote the first-stage constraints (1b)–(1i) and
(4a)–(4c) by the feasibility set $X_1$. We then formulate a relaxed master problem in the first stage as

$$\text{MP} : \min c^T x + \theta$$

s.t. \((x, y, u, r, z) \in X_1\)

$$C(u, r, z, \theta) \geq 0$$

$$\theta \in \mathbb{R}$$

where $\theta$ denotes an estimated lower bound for $\sum_{\omega \in \Omega} p^\omega Q(u, r, z, \omega)$, and $C(u, r, z, \theta) \geq 0$ is the set of optimality cuts derived from second-stage subproblems, which consists of linear functions.

We notice here that the feasibility set composed by constraints (3a)–(3f) is always nonempty with any first-stage decisions \((u, r, z)\). Therefore, we do not need any feasibility cuts.

In the second-stage problem (3), we associate dual variables $\gamma_j$ to constraints (3b), $\pi^j_k$ to constraints (3c), $\lambda_j$ to constraints (3d), and $\alpha_j$ to constraints (3e), and then the dual program can be formulated as $Q(u, r, z, \omega) =$

$$\max \sum_{j \in J} \gamma_j \left( \sum_{i \in I} t_{\text{depot}, i}(\omega) u^j_{i,1} - r^j_1 \right)$$

$$+ \sum_{k=1}^{|K|-1} \sum_{j \in J} \pi^j_k \left( r^j_k + \sum_{i \in I} q_i(\omega) T_i(\omega) u^j_{i,k} + \sum_{i_1, i_2 \in \{ \text{depot} \}} t_{i_1, i_2}(\omega) u^j_{i_1,k} u^j_{i_2,k+1} - r^j_{k+1} \right)$$

$$+ \sum_{j \in J} \lambda_j \left( r^j_{|K|} + \sum_{i \in I} q_i(\omega) T_i(\omega) u^j_{i,|K|} + \sum_{i \in I} t_{\text{depot}, i}(\omega) u^j_{i,|K|} - L_j \right)$$

$$+ \sum_{j \in J} \alpha_j \left( \sum_{k \in K} \sum_{i \in I} q_i(\omega) \tau_{O_i, D_i}(\omega) u^j_{i,k} + \sum_{i \in I} t_{\text{depot}, i}(\omega) u^j_{i,1} \right)$$

$$+ \sum_{k=1}^{|K|-1} \sum_{i_1, i_2 \in \{ \text{depot} \}} t_{i_1, i_2}(\omega) u^j_{i_1,k} u^j_{i_2,k+1} + \sum_{i \in I} t_{i, \text{depot}}(\omega) u^j_{i,K}$$

s.t. $\gamma_j - \pi^j_1 \leq c^w$, $\forall j \in J$ \hspace{1cm} (11a)

$$\pi^j_{k-1} - \pi^j_k \leq c^w$, $\forall k = 2, \ldots, |K| - 1, \forall j \in J$ \hspace{1cm} (11b)

$$\pi^j_{|K|-1} - \lambda_j \leq c^w$, $\forall j \in J$ \hspace{1cm} (11c)

$$-\gamma_j \leq c^l$, $\forall j \in J$ \hspace{1cm} (11d)

$$-\pi^j_{k-1} \leq c^l$, $\forall k = 1, \ldots, |K| - 1, \ j \in J$ \hspace{1cm} (11e)

$$-\lambda_j \leq c^l$, $\forall j \in J$ \hspace{1cm} (11f)

$$\lambda_j \leq c^o$, $\forall j \in J$ \hspace{1cm} (11g)
\[ \alpha_j \leq c^d, \quad \forall j \in J, \]  

where we denote the constraints (11b)–(11i) by the feasibility set \( X_2 \), and denote the objective (11a) as \( f(\gamma, \pi, \lambda, \alpha, u, r, z, \omega) \) for notation simplicity.

We rewrite the second-stage subproblem as

\[
\text{SP}^w(u, r, z) : \max f(\gamma, \pi, \lambda, \alpha, u, r, z, \omega) \\
\text{s.t.} \quad (\gamma, \pi, \lambda, \alpha) \in X_2.
\]

Then the procedure of Benders’ decomposition algorithm can be described in Algorithm 4.

**Algorithm 4** Benders’ decomposition to solve TS-MILP

1: Let iteration \( l = 1 \), and initialize MP without any cuts \( C(u, r, z, \theta) \geq 0 \).
2: while some termination criteria is not satisfied do
3: \quad Solve MP, and obtain an optimal solution \((x^l, y^l, u^l, r^l, z^l, \theta^l)\). (When there are no cuts, let \( \theta^l = -\infty \))
4: \quad for \( \omega \in \Omega \) do
5: \quad \quad Solve SP^w(u, r, z) and denote \((\gamma(\omega)^l, \pi(\omega)^l, \lambda(\omega)^l, \alpha(\omega)^l)\) as an optimal solution.
6: \quad end for
7: \quad if \( \theta^l \geq \sum_{\omega \in \Omega} p^w(f(\gamma(\omega)^l, \pi(\omega)^l, \lambda(\omega)^l, \alpha(\omega)^l, u^l, r^l, z^l, \omega)) \) then
8: \quad \quad Stop and claim optimality of solution \((x^l, y^l, u^l, r^l, z^l, \theta^l)\)
9: \quad else
10: \quad \quad Generate cut \( \theta \geq \sum_{\omega \in \Omega} p^w(f(\gamma(\omega)^l, \pi(\omega)^l, \lambda(\omega)^l, \alpha(\omega)^l, u, r, z, \omega)) \) into cut set \( C(u, r, z, \theta) \).
11: \quad end if
12: \quad \( l := l + 1 \)
13: end while

**C** Other Results of K-means-based Heuristics

In this section, we present other out-of-sample results of K-means-based heuristics under different settings, where we vary the number of swapping steps \( \ell_{\text{max}} \) from 5 to 15 in Table 15, and change the input feature matrix \( \{d_i\}_{i=1}^{\lvert I \rvert} \) in Table 16. We use KM-duration, KMS-duration, KMSS-duration to denote the corresponding variants of adding empirical service duration to the input feature matrix. From both tables, these elements do not turn out to be main factors to improve the performance. Notably, adding service duration to the input feature matrix even negatively impacts the overall objective cost in KMS and KMSS, as it tends to group customers having large service duration together, while leaving other customers in the same vehicle.
Table 15: Out-of-sample results of K-means with varying swapping step $\ell_{\text{max}}$

| $|J|$ | $|I|$ | $\ell_{\text{max}}$ | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. (\$) | Time (sec.) |
|-----|-----|----------|---------|---------|---------|---------|---------|----------|
| 5   | 4   | 434.65   | 0.12    | 0.00    | 207.14  | 2027.80 | 0.13    |
| 16  | 3   | 10       | 434.80  | 0.12    | 0.00    | 207.62  | 2028.24 | 0.16    |
| 15  |     | 10       | 434.80  | 0.12    | 0.00    | 207.62  | 2028.24 | 0.14    |

Table 16: Out-of-sample results of K-means with varying input feature matrix $\{d_{ij}\}_{i=1}^{|I|}$

| $|J|$ | $|I|$ | Method         | ID (min.) | WT (min.) | OT (min.) | TT (min.) | Obj. (\$) | Time (sec.) |
|-----|-----|----------------|---------|---------|---------|---------|---------|----------|
| 16  | 3   | KM             | 435.40  | 2.22    | 0.24    | 203.52  | 2104.54 | 0.13    |
|     |     | KM-duration    | 418.71  | 3.33    | 0.09    | 209.50  | 2085.33 | 0.15    |
|     |     | KMS            | 434.65  | 0.12    | 0.00    | 207.14  | 2027.80 | 0.13    |
|     |     | KMS-duration   | 439.94  | 0.19    | 0.05    | 209.44  | 2047.54 | 0.13    |
|     |     | KMSS           | 404.92  | 0.10    | 0.00    | 207.14  | 1937.87 | 5.19    |
|     |     | KMSS-duration  | 411.00  | 0.08    | 0.00    | 209.50  | 1955.42 | 5.59    |