A Dynamic Routing Framework for Shared Mobility Services

YUE GUAN and ANURADHA M. ANNASWAMY, Massachusetts Institute of Technology, USA
H. ERIC TSENG, Ford Motor Company, USA

Travel time in urban centers is a significant contributor to the quality of living of its citizens. Mobility on Demand (MoD) services such as Uber and Lyft have revolutionized the transportation infrastructure, enabling new solutions for passengers. Shared MoD services have shown that a continuum of solutions can be provided between the traditional private transport for an individual and the public mass transit-based transport, by making use of the underlying cyber-physical substrate that provides advanced, distributed, and networked computational and communicational support. In this article, we propose a novel shared mobility service using a dynamic framework. This framework generates a dynamic route for multi-passenger transport, optimized to reduce time costs for both the shuttle and the passengers and is designed using a new concept of a space window. This concept introduces a degree of freedom that helps reduce the cost of the system involved in designing the optimal route. A specific algorithm based on the Alternating Minimization approach is proposed. Its analytical properties are characterized. Detailed computational experiments are carried out to demonstrate the advantages of the proposed approach and are shown to result in an order of magnitude improvement in the computational efficiency with minimal optimality gap when compared to a standard Mixed Integer Quadratically Constrained Programming-based algorithm.

CCS Concepts: • **Applied computing** → **Transportation**; • **Mathematics of computing** → *Paths and connectivity problems*; • **Computing methodologies** → *Modeling methodologies*;

Additional Key Words and Phrases: Smart cities, shared mobility, mobility on demand, dynamic routing, alternating minimization, mixed integer quadratically constrained programming, space window, clustering

ACM Reference format:

Yue Guan, Anuradha M. Annaswamy, and H. Eric Tseng. 2019. A Dynamic Routing Framework for Shared Mobility Services. *ACM Trans. Cyber-Phys. Syst.* 4, 1, Article 6 (October 2019), 28 pages.
<https://doi.org/10.1145/3300181>

NOMENCLATURE

$\alpha_1, \alpha_2, \alpha_3^p, \alpha_3^d$	weights of travel time cost terms defined in the objective function
$\bar{h}$	number of iterations before termination defined in the AltMin algorithm
γ_1, γ_2	weights of travel time cost terms defined in the objective function
C_{AltMin}^*	optimal cost derived via the AltMin algorithm
C_{MIQCP}^*	optimal cost derived via the MIQCP based approach

This work was supported by the Ford-MIT Alliance.

Authors' addresses: Y. Guan (corresponding author) and A. M. Annaswamy, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, 02139, USA; emails: {guany, aanna}@mit.edu; H. E. Tseng, Ford Motor Company, 1 American Road, Dearborn, MI, 48126, USA; email: htseng@ford.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

2378-962X/2019/10-ART6 \$15.00

<https://doi.org/10.1145/3300181>

C	objective function
$\mathcal{E}$	entropy of the state defined in the modified BHK algorithm
A_i	area to visit of cluster C_i
c	capacity of the shuttle
C^*, S^*, R^*	optimal cost, corresponding sequence, and routing points, respectively
C_i	i th cluster
cl_k^p, cl_k^d	indexes of clusters containing E_k^p and E_k^d , respectively
$d(\cdot, \cdot)$	distance metric
e_i, e'_i	indicators that whether A_i has been visited at the current and possible next states defined in the modified BHK algorithm, respectively
E_k^p, E_k^d	events of passenger k being picked up and dropped off, respectively
f_i^p, f_i^d	numbers of finished pickups and drop-offs before the shuttle visits A_i , respectively
F_{ij}^p, F_{ij}^d, F_c	intermediate variables defined to linearize constraints (19) to (31), and (20) to (32)
g_i^1, g_i^2	numbers of passengers not picked up, and on board when the shuttle is travelling on the i th trip segment, respectively
h	index of iteration defined in the AltMin algorithm
h^*	index of the iteration corresponding to the optimal cost defined in the AltMin algorithm
h_0, h'_0	indexes of iterations between which the AltMin algorithm recurs, $h'_0 < h_0$
$h_{\max}$	upper bound on the number of iterations defined in the AltMin algorithm
i, j, k	indexes for ease of notation, i, j for clusters, and k for passengers
L, L'	indexes of the current and next possible clusters that the shuttle serves defined in the modified BHK algorithm, respectively
l_i^p, l_i^d, l_i	numbers of pickups, drop-offs, and net loads of C_i
M	multiplier of $t(L, L')$ contributing to C defined in the modified BHK algorithm
MPS^p, MPS^d	upper bounds on the pickup and drop-off position shifts, respectively
N	numbers of clusters
n	numbers of passengers
O	origin of the shuttle
P_k, D_k	requested pickup and drop-off locations of passenger k , respectively
p_k, d_k	orders of passenger k in the pickup and drop-off queues, respectively
q_i	number of passengers on board after the shuttle departs from R_i
Q_{ij}, Q_c	intermediate variables defined to linearize constraint (18) to (30)
R, R_f	routing points of the clusters, and the feasible domain of R , respectively
R_i	routing point of cluster C_i
r_k^p, r_k^d	requested maximum walking distances before being picked up, and after being dropped off of passenger k , respectively
$R_{ij}^r, R_k^{s,p}, R_k^{s,d}, R_i^r$	explaining decision variables for convex relaxation via SOC constraints defined in Equations (33), (34), (45), and (46)
$RideT_k$	riding time of passenger k
S, S_f	sequence of the clusters, and the feasible domain of S , respectively
S^h, R^h	S and R after the h th iteration defined in the AltMin algorithm, respectively
$S_W(P_k, r_k^p)$	space window of locations with distances from P_k not exceeding r_k^p
$t(L, L')$	time taken for the shuttle to travel from A_L to $A_{L'}$ defined in the modified BHK algorithm
T_i	departure time of the shuttle from R_i
t_i	travel time of the shuttle on the i th trip segment

T_k^r	time of request of passenger k
t_{ij}	travel time of the shuttle from R_i to R_j
T_{ij}, T_c	intermediate variables defined to linearize constraint (15) to (29)
V	value function of the state defined in the modified BHK algorithm
v_p	walking speed of the passengers
v_s, a, t_s, t_a	constants of the shuttle travel mode
w_k^p, w_k^d	walking times before being picked up, and after being dropped off of passenger k , respectively
$WaitT_k$	waiting time of passenger k
$Walk_k^p, Walk_k^d$	walking times before being picked up, and after being dropped off of passenger k , respectively
x_{ij}	indicator that whether the shuttle travels from A_i to A_j
Z_1, Z_2, Z_3, Z_4	sets defined for ease of notation on pages 7 and 11

1 INTRODUCTION

Mobility of people and goods has been critical to urban life ever since cities emerged thousands of years ago. Over 1 billion vehicles travel on the roads today, and that number is projected to double by 2050 [14]. New approaches and solutions are required to solve, or at least improve the quality of urban mobility, both on highways and on city streets. Until recently, available solutions for urban transportation have been clearly binary, with the first option represented by public transportation that provides low cost solutions and reduced carbon footprint per traveler at increased times of walking, waiting, and traveling, and the second represented by individual private automobiles that reduce individuals' times but with a significantly high cost and increased carbon footprint. With the ushering in Cyber-Physical Systems enabled by the development of smart mobile devices as well as affordable, accessible and powerful computing resources, a novel mode of urban mobility named Mobility on Demand (MoD) services has been revolutionizing the ground transportation infrastructure in urban centers [5, 10, 22]. Compelling statistics such as the idle rate of vehicles in the United States and United Kingdom being at 95% [1] has enabled the emergence of several MoD companies worldwide, such as Uber, Lyft, and Didi Chuxing, thereby enabling new solutions for passengers, with flexibility in travel time and cost.

MoD solutions of late have started to provide several alternatives other than conventional transportation options, e.g., buses, subways and taxis, for passengers to move around. These correspond to different combinations of travel time, cost, and carbon footprint per traveler there by introduce different and attractive alternatives to passengers [20, 32]. A recent article [4] illustrates clearly that the potential of shared mobility in drastically reducing the number of vehicles on the roads is very high. The article demonstrates that if ride sharing is implemented in New York City, 98% of the taxi demands could be satisfied by 3,000 four-passenger cars, which is fewer than 25% of the number of taxis currently operating in New York City with a waiting time of 2.7 minutes on average. Furthermore, over 95% of the trips are feasible to be shared with other trips given a maximum 5-minute travel delay [26]. Similar empirical laws appear to be applicable in other cities as well, suggesting that the potential of shared mobility solutions is a global one [28]. This suggests that customized shared MoD solutions that involve multi-passenger transport can provide an ideal combination of convenience, flexibility, and affordability to passengers (Figure 1) [2, 6, 16]. In this article, we present one such shared MoD solution that uses a dynamic routing framework for operating a multi-passenger shuttle.

The underlying goal in a shared mobility solution is the reduction of Price of Anarchy (PoA), which can be loosely defined as the cost ratio of all agents acting in a decentralized manner and that

Fig. 1. Spectrum of mobility services.

due to a completely centralized infrastructure [31]. Our thesis is that our proposed shared MoD service can achieve this reduction by employing a dynamic framework, which responds to real time travel needs of passengers and incentivized to ensure their subscriptions to these routes. The specific dynamic framework that we propose is a dynamic shuttle service that responds to real time requests from passengers, determines a route that is customized to their requested pickup/drop-off locations, which optimizes a time cost that includes the total travel time of the shuttles as well as time costs incurred by the passengers. While this framework is also capable of incorporating dynamic prices for the shuttle rides [6], we focus in this article only on the dynamic routing of the shuttle.

Dynamic routing is a main feature of ride sharing services. Ride sharing has been the subject of study in many papers, including References [4, 7, 18, 26–28]. Reference [27] is a macroscopic study of the Singapore mobility scene and estimates the financial benefits of ride sharing. References [4, 26] focus on the simulation of actual implementations of ride sharing, with the latter providing a more elaborate case study. Reference [18] discusses ride sharing using a case study of Paris and the underlying ride sharing strategies. Reference [33] proposes a transit service to tackle the last-mile problem and validates its capability of reducing last-mile distances and travel times via large-scale simulation based upon real data. What makes our article distinct and novel compared to these other works is the use of a new concept that we have introduced, termed as *space window*. While the concept of dynamic routing is considered in many of these papers, none of them considers the added flexibility and efficiency in routing provided by space windows.

The main idea of a space window is that the passengers are willing to walk for a certain distance, both before being picked up and after being dropped off to get the dynamic shuttle service. Passengers specify the maximum distances they are willing to walk, and therefore, the requested pickup/drop-off locations expand to pickup/drop-off regions denoted as space windows S_W , which are defined as the sets of all feasible locations that passengers are willing to walk to/from. Space windows are one of the key ingredients of the proposed dynamic shuttle service, which can, therefore, be viewed as a semi door-to-door mode (Figure 2). Our thesis is that the introduction of S_W provides greater flexibility and an important degree of freedom in reducing PoA. Moreover, as will be seen in the sections that follow, the introduction of S_W allows a clustering of different pickup/drop-off events, which is able to reduce the number of shuttle stops and in turn leads to a

Fig. 2. Complete door-to-door mode versus semi door-to-door mode.

significant reduction in the time cost compared to a traditional MoD service. Further benefits are due to real street topologies and traffic congestion and the associated asymmetries between driving time and walking time, such as one-ways and pedestrian only lanes as a short walking distance for a passenger may result in significant improvements in the shuttle route. Such an increase in the quality of service has the potential to provide additional positive externalities on the customer riding experiences as well. An elaborate comparison of dynamic routing with space windows versus that without space windows has been conducted in Reference [6] using real operational data, demonstrating an average improvement of 30% reduction on travel time costs and 50% reduction on the number of shuttle stops. However, space windows also introduce penalties in terms of extra walking times for passengers, leading to a trade-off in the shuttle service design. The maximum distance willing to walk is not only the protocol between a passenger and the shuttle server when negotiating a ride but also a key tuning parameter to adjust the trade-off. One should also accommodate the fact that different customers may have different preferences in terms of how far they are willing to walk, or the same passenger may have different preferences depending on various factors including weather conditions, seasons, and times of day.

With the above concept of a space window, the overall shared mobility service is proposed to operate in four steps: (1) *Request*: passengers request shuttle rides with specified pickup/drop-off locations, maximum distances willing to walk, and time windows of service if needed. (2) *Offer*: the shuttle server distributes offers to passengers with ride details including pickup locations, walking distances, pickup times, drop-off locations, drop-off times, and prices. (3) *Decide*: passengers decide whether to accept or decline the offers. (4) *Operate*: the shuttle server sends out operational instructions to the shuttles according to the decisions from the passengers and trips start.

The introduction of space windows, however, introduces significant challenges in designing an analytically tractable optimization framework. The overall problem of conventional dynamic routing with complete door-to-door mode, often referred to as a Dial-a-Ride Problem (DARP) has been studied extensively in the literatures. The underlying problem is essentially a Traveling Salesman Problem (TSP). The concept of space windows introduces extra complexity to the optimization framework, where dynamic routing with the therefore semi-door-to-door mode is analogous to a TSP between regions as opposed to points. Though the overall problem can be formulated as a standard Mixed Integer Programming (MIP), it is extremely computationally burdensome. In this article, we propose the use of an Alternating Minimization-based [8, 19, 24] optimization approach to tackle the problem. In addition, we use this approach to carry out dynamic routing in response to real time requests from passengers, thereby facilitates an overall dynamic framework. This framework not only includes dynamic routing, which is the focus of this article, but also dynamic pricing [6].

Fig. 3. Procedures of the dynamic shuttle service: request, offer, decide, and operate.

In summary, the main contributions of this article are the following: (i) a novel dynamic routing framework for shared mobility services, which utilizes multi-passenger shuttles, built upon a central concept termed as space window; (ii) a constrained optimization formulation that shows how the space window concept can be utilized to carry out dynamic routing of shuttles, delineated by first carrying out static routing, for ease of exposition, and then the dynamic case; (iii) an Alternating Minimization (AltMin) algorithm to solve the constrained optimization problem, benchmarked against a standard Mixed Integer Quadratically Constrained Programming (MIQCP)-based approach; (iv) quantifications of the advantages of the proposed AltMin algorithm using real operational data. As will be shown in the sections that follow, the proposed framework is able to reduce the number of shuttle stops and further save travel times for both the shuttle and the passengers. The AltMin algorithm is demonstrated to achieve an order of magnitude improvement in the computational efficiency and preserves optimality compared to the MIQCP-based approach in various scenarios.

The overall organization of the article is as follows. We formally state the problem of a shared mobility service with space window under a static routing setup for a single shuttle case in Section 2, using a constrained optimization formulation. Before proceeding to detail the AltMin algorithm we propose to solve static routing, we describe the standard MIQCP formulation in Section 3, which will be used for the benchmark. In Section 4, we describe the AltMin algorithm. Detailed computational experiments are carried out in Section 5, using real operational data from References [3, 16], to evaluate the performances of the AltMin algorithm, and compare closely with the MIQCP-based approach. Extensions to dynamic routing using its static counterpart as a subcomponent are discussed in Section 6. A summary and concluding remarks are provided in Section 7.

2 PROBLEM FORMULATION

In this section, we formulate the problem that we propose to address in the context of shared mobility services. The goal is to generate a route for a multi-passenger shuttle to optimize the underlying time costs. In this section, we restrict our attention to static routing, where the assumption is that all ride requests have arrived prior to the departure of the shuttle, and once the trip starts, no further requests will be accepted until the current rides are completed. The overall goal of the shared mobility service is the generation of a static route using the four steps described in Figure 3 in Section 1, which will be shown in this section, to be in the form of a constrained optimization problem. The static route that is proposed by the server is determined to optimize various time costs that are incurred both by the shuttle as well as by the passengers. In the former

case, we include the shuttle travel time, and in the latter case, we include the walking times, waiting times, and ride times incurred by the passengers. Before we proceed to describe the algorithms, we first present a few preliminaries in Section 2.1. In Section 2.2, we formally state the objective function, and in Section 2.3, we delineate various constraints. The overall *master problem* is then stated formally in Section 2.4.

2.1 Preliminaries

Definition 2.1 (Ride Request). A *ride request* of the dynamic shuttle service is defined as a tuple $\{P_k, D_k, r_k^p, r_k^d, T_k^r\}$, composed of P_k the customer specified pickup location, D_k drop-off location, r_k^p and r_k^d the maximum distances willing to walk before being picked up and after being dropped off, respectively, as well as T_k^r the time of request. $P_k, D_k \in \mathbb{R}^2$ (e.g., longitude and latitude), $r_k^p, r_k^d \in \mathbb{R}_{\geq 0}$, and $T_k^r \in \mathbb{R}_{\geq 0}$. Let $Z_1 = \{1, 2, \dots, n\}$, where $n \in \mathbb{Z}_{>0}$ is the number of requests. $k \in Z_1$ denotes the order of the request according to the time stamp when it is made, i.e., a smaller k value corresponds to an earlier request. Without loss of generality, we assume there is only one passenger per request, and all the requests are made at time $T_k^r = 0, \forall k \in Z_1$.

Definition 2.2 (Space Window). A pickup/drop-off *space window* is defined as the set of all feasible pickup/drop-off locations whose distance from the requested pickup/drop-off location does not exceed the specified maximum walking distance. For each $k \in Z_1$, denote $S_W(P_k, r_k^p), S_W(D_k, r_k^d) \subset \mathbb{R}^2$ as the pickup and drop-off space window for the k th passenger, respectively, such that

$$S_W(P_k, r_k^p) = \{x \mid x \in \mathbb{R}^2, d(x, P_k) \leq r_k^p\}, \quad (1)$$

$$S_W(D_k, r_k^d) = \{x \mid x \in \mathbb{R}^2, d(x, D_k) \leq r_k^d\}, \quad (2)$$

where d is the distance metric on $\mathbb{R}^2$.

Let E_k^p and E_k^d denote the events that correspond to the k th passenger being picked up and dropped off, respectively. If during the same trip the space windows of different pickup/drop-off events share a common intersection, then the corresponding events are assumed eligible to be grouped. In such a case, the shuttle is allowed to visit the intersection to take care of all the corresponding pickup/drop-off tasks simultaneously, instead of visiting individual space windows and serving the tasks one by one.

Definition 2.3 (Cluster). A *cluster* C_i is a nonempty set of pickup and/or drop-off events if the corresponding space windows share a common intersection, where $i \in \mathbb{Z}_{>0}$ is the index of the cluster.

It should be noted that the existence of a shared intersection is necessary for clustering, but not sufficient. This will be discussed in Section 2.4.

Definition 2.4 (Clustering Pattern). A *clustering pattern* is a set of mutually exclusive clusters $\{C_1, C_2, \dots, C_N\}$, whose union constitutes the set of all pickup and drop-off events, where $N \in \mathbb{Z}_{>0}$ denotes the number of clusters.

Let $Z_2 = \{1, 2, \dots, N\}$. By definition, $\bigcup_{i \in Z_2} C_i = \{E_k^p, E_k^d, \forall k \in Z_1\}$. Also, $C_i \cap C_j = \emptyset, \forall i, j \in Z_2, i \neq j$. $N = 2n$ corresponds to the case where no nontrivial clusters are formed, that is, each pickup/drop-off event itself is a cluster. Typically, $N < 2n$, because the likelihood that different events are eligible to be clustered is fairly high for peak travel conditions in urban areas. It should be noted that some clusters may only contain pickups or drop-offs, and some may contain both.

Definition 2.5 (Area). An *area* $A_i, \forall i \in Z_2$ is defined as a subset of $\mathbb{R}^2$ that represents the feasible area to visit associated with the cluster C_i .

It follows that A_i is the intersection of the corresponding space windows of the events contained in C_i , and is given by

$$A_i = \{x \mid x \in S_W(P_{k_p}, r_{k_p}^p) \cap S_W(D_{k_d}, r_{k_d}^d), \forall E_{k_p}^p, E_{k_d}^d \in C_i\}. \quad (3)$$

Definition 2.6 (Routing Point). A routing point $R_i \in A_i, \forall i \in Z_2$ is defined as the actual location that the shuttle visits in area A_i to pick up and/or drop off the corresponding passengers.

For the sake of consistency, throughout this manuscript, Z_1 and Z_2 will be used as the sets that contain the indexes of the passengers (or requests) and clusters, respectively. Moreover, subscripts $i, j \in Z_2$ are associated with clusters, and subscript $k \in Z_1$ is associated with passengers.

The routing task of the dynamic shuttle service is essentially to design a route to serve a given set of ride requests, which minimizes the cost of the shuttle-passenger system and satisfies certain service constraints, which are defined as follows.

2.2 Objective Function

The objective function of the system is defined as follows:

$$C = \gamma_1 \sum_{i=1}^N t_i + \gamma_2 \sum_{k=1}^n (\alpha_1 \text{Wait}T_k + \alpha_2 \text{Ride}T_k + \alpha_3^p \text{Walk}T_k^p + \alpha_3^d \text{Walk}T_k^d), \quad (4)$$

which represents a weighted sum of various time costs, both on the shuttle side and on the passenger side. In Equation (4), $\forall i \in Z_2$, t_i denotes the time taken by the shuttle to complete the i th trip segment, $\forall k \in Z_1$, $\text{Wait}T_k$, $\text{Ride}T_k$, $\text{Walk}T_k^p$, and $\text{Walk}T_k^d$ denote the waiting time, riding time, walking time before being picked up and after being dropped off of passenger k , respectively, and $\gamma_1, \gamma_2, \alpha_1, \alpha_2, \alpha_3^p$, and α_3^d are nonnegative weights.

It should be noted that the objective function defined in Equation (4) is a general representation, which does not depend on any specific distance metric, i.e., versatile for various distance metrics as long as represented properly. Throughout this manuscript, the underlying space topology is assumed to be a two dimensional Euclidean space and as a result the corresponding space windows $S_w(P_k, r_k^p)/S_w(D_k, r_k^d)$ are circles centered at P_k/D_k with r_k^p/r_k^d as the radii, $\forall k \in Z_1$. A similar formulation can be extended to other metrics as well, such as the Manhattan distance and metrics that correspond to real street topologies.

As has been briefly discussed in Section 1, the reduction on N has potential to facilitate the reduction on C , especially when considering the actual travel profile of the shuttle. For each $i \in Z_2$, the term t_i includes both the actual travel time on the i th trip segment and the service time at the stop. The travel time is calculated assuming a constant shuttle velocity v_s . However, in reality, the shuttle accelerates and decelerates between stops. Therefore, an extra amount of time $\frac{v_s}{2a}$, which is a reasonable approximation of the average additional time taken considering varying velocity profile, is added for correction. a denotes the shuttle acceleration. Moreover, the shuttle stops at each station for a service time t_s to serve the passengers. Hence, t_i is calculated as the sum of the travel time using constant velocity, and augmented with an additional time t_a given by

$$t_a = t_s + \frac{v_s}{2a}. \quad (5)$$

Therefore, our formulation applies implicit penalty on N , as t_a accumulates when the shuttle travels along the route.

2.3 Constraints

The route of the dynamic shuttle service should satisfy the following constraints.

Definition 2.7 (Legitimate Constraint). *Legitimate constraints* specify that any passenger should be picked up before being dropped off, that being said, $\forall k \in Z_1$, E_k^p should occur before E_k^d . Moreover, E_k^p and E_k^d cannot be in the same cluster as otherwise passenger k would be willing to walk directly from the requested origin to the destination instead of taking a shuttle ride.

Definition 2.8 (Capacity Constraint). *Capacity constraints* specify that at anytime of the trip, the number of passengers on board should not exceed the capacity of the shuttle denoted as $c \in \mathbb{Z}_{>0}$.

It should be noted that the number of passengers within a trip could be larger than c , as long as they are not on board simultaneously at anytime during the trip.

Definition 2.9 (Maximum Position Shift Constraint). A pickup/drop-off *position shift* is defined as the absolute value of the difference between the actual occurrence of the pickup/drop-off event and that of the request event. *Maximum Position Shift (MPS) constraints* specify that the position shifts should not exceed some given upper bounds, that being said,

$$|p_k - k| \leq MPS^p, \quad \forall k \in Z_1, \quad (6)$$

$$|d_k - k| \leq MPS^d, \quad \forall k \in Z_1, \quad (7)$$

where p_k and d_k denote the orders of the k th passenger in the pickup and drop-off queue, respectively. MPS^p and MPS^d are positive integer upper bounds on the pickup and drop-off position shifts, respectively, which are provided by the dynamic shuttle server as part of the service protocol.

MPS constraints are priority constraints that serve as a guarantee of the dynamic shuttle service quality, such that passengers will not experience a long waiting time or travel delay. They are especially significant in the dynamic routing scenario to prevent the possibility of any passenger from indefinite deferment of pickup and drop-off by the algorithm.

From the above definitions, we note that there are essentially three queues associated with the passengers, which correspond to three different sequences $\{k\}$, $\{p_k\}$, and $\{d_k\}$, where $\{k\}$ corresponds to the sequence of requests, $\{p_k\}$ and $\{d_k\}$ correspond to those of pickups and drop-offs, respectively. The sequences $\{p_k\}$ and $\{d_k\}$ are permutations of $\{k\}$ in the extreme case where each cluster consists of only one event. In general, $\{p_k\}$ and $\{d_k\}$ include both permutations of subsets of $\{k\}$ and repetitions, as the orders of different pickups/drop-offs may be tied if in the same cluster.

Definition 2.10 (Departure Constraint). *Departure constraints* specify that for those clusters that contain pickup events, the shuttle should not depart from the routing point before the associated passengers arrive and get on board.

In our problem formulation, legitimate, capacity, and MPS constraints are named *primary constraints* as they affect the feasibility of the clustering pattern and the sequence of the clusters to be visited. While the departure constraints are called *secondary constraints*, which can always be satisfied by means of forcing the shuttle to wait at the routing points and adjust the departure times, as long as the three sets of primary constraints have all been met.

2.4 Master Problem

To serve a given set of requests, the dynamic shuttle service needs to determine: (i) the clustering pattern, (ii) the sequence of clusters to be visited, and (iii) the routing point of each cluster. Then the shuttle will start from the depot O , visit each routing point exactly once in the right order, and

Fig. 4. Scenarios where certain favorable sequences or routing points are ruled out due to clustering. (a) The blue route outperforms the red one, which is the only feasible route if cluster $\{E_2^p, E_1^d\}$, $\{E_3^p, E_2^d\}$, and $\{E_4^p, E_3^d\}$, in terms of the total travel distance of the shuttle. (b) The route on the right is worse in terms of the total travel distance, if $\{E_2^p, E_1^d\}$ are grouped; therefore, there is only one feasible routing point for the cluster.

end at a feasible terminal point. In this manuscript, we will not be covering how to determine the clustering pattern, but rather briefly address two main features while designing clusters as follows:

- (1) A clustering pattern should be *feasible*.

A clustering pattern is feasible if and only if there exists at least one feasible sequence of the corresponding clusters that satisfies all the three sets of primary constraints. It should be noted that the primary constraints are originally defined for individual passengers, once clusters are formed, they are transferred to clusters, i.e., sets of events associated with individual passengers. For example, legitimate constraints are transferred in the way that $\forall k \in Z_1$, the cluster that contains E_k^p should be visited before that contains E_k^d . Capacity and MPS constraints can be transferred as well. Feasibility is a *hard requirement* for clustering patterns.

- (2) A clustering pattern should be *favorable*.

Feasibility is necessary for a clustering pattern, however, not sufficient. The reason is that though clustering helps reduce the number of stops for a trip serving a given set of passengers, which is likely to reduce the cost of the system, it may also possibly rule out certain favorable sequences or routing points due to extra restrictions resulted from the binding of different events (Figure 4). Therefore, a good clustering pattern that can be in use should also be favorable in the sense that it has potential to result in a reduction on C defined in (4), through a suitable design of the sequence and routing points. Though hard to define in a rigorous manner, the main philosophy behind a favorable clustering pattern is that it groups events that are not only close in space, i.e., share common intersections, but also close in the optimal sequence if no clustering is applied. Favorability is a *soft requirement* for clustering patterns.

In what follows, we will assume that a feasible and favorable clustering pattern is explicitly given, based on which the determination of the sequence and routing points will be explored to minimize the cost defined in Equation (4). To formally state this optimization problem, we construct a N dimensional vector S to represent the sequence of the clusters to be visited, where S is a permutation of $[1, 2, \dots, N]$, $S(i)$ indicates the order of cluster C_i in the visiting queue of the shuttle, $\forall i \in Z_2$. Denote $C_0 = \emptyset$ as a dummy cluster representing the depot, whose corresponding space window is $S_W(O, 0)$ and $A_0 = \{O\}$, $R_0 = O$. Denote $R = (R_0, R_1, \dots, R_N)$. With the objective function specified as in Equation (4), and the four sets of constraints as in Definitions 2.7 through

2.10, the *master problem* for the rest of the manuscript is formally stated as

$$\min_{(S,R) \in S_f \times R_f} C(S,R), \quad (8)$$

where S_f and R_f are the feasible domains of the decision variables S and R , respectively, that satisfy all of the constraints listed in Section 2.3, and $R_f = A_0 \times A_1 \times \dots \times A_N$.

3 A MIXED INTEGER QUADRATICALLY CONSTRAINED PROGRAMMING FORMULATION

The underlying master problem is to derive the optimal route, i.e., sequence and routing points, that minimizes the time cost of the system defined in Equation (4) subject to the four sets of constraints specified in Definitions 2.7–2.10. This has been compactly restated as the optimization problem in Equation (8). Given the discrete nature of S and the continuous nature of R , it follows that the problem is one of a MIP. As the Euclidean distance metric is exploited, the formulation includes quadratic constraints as well. A natural next step, therefore, is to evaluate the use of a standard MIQCP formulation to solve the master problem. An MIQCP formulation is a model of an optimization problem that exploits both continuous and discrete decision variables and has constraints containing quadratic terms [9]. In this section, we carry out such an evaluation.

Given that the shuttle starts the trip from the depot O and ends at a terminal cluster, in addition to C_0 denoting the depot, we add another dummy cluster C_{N+1} to denote the virtual last stop. This is to ensure that the problem has an open-ended setting so that the shuttle is eligible to terminate the trip at any cluster as long as it is feasible. The travel time from the routing point of any other cluster to that of C_{N+1} is set to be zero and the shuttle is forced to end the trip with C_{N+1} . The associated A_{N+1} and R_{N+1} have no actual physical meaning.

In addition to Z_1 and Z_2 , we introduce Z_3 and Z_4 to facilitate the formulation as follows:

- (1) $Z_3 := \{(i,j) \mid i \in \{0\} \cup Z_2, j \in Z_2 \cup \{N+1\}, i \neq j, \text{ and } (i,j) \neq (0, N+1)\}$,
- (2) $Z_4 := \{(i,j) \mid i \in \{0\} \cup Z_2, j \in Z_2, \text{ and } i \neq j\}$.

For each pair of clusters $(i,j) \in Z_3$, let $x_{ij} = 1$ if the shuttle travels from A_i to A_j , otherwise $x_{ij} = 0$. t_{ij} denotes the travel time of the shuttle from R_i to R_j . For each $i \in \{0\} \cup Z_2 \cup \{N+1\}$, T_i denotes the departure time of the shuttle from R_i , and $T_0 = 0$ is set as the departure time of the trip from O . q_i denotes the number of passengers on board after the shuttle visits cluster C_i . It is clear that $q_0 = q_{N+1} = 0$. f_i^p and f_i^d denote the number of finished pickups and drop-offs before the shuttle visits cluster C_i , respectively. It follows that $f_0^d = f_0^p = 0$ and $f_{N+1}^p = f_{N+1}^d = n$. Let ℓ_i^p , ℓ_i^d and ℓ_i denote the number of pickups, drop-offs and net loads of cluster C_i , respectively, where net load denotes the difference between the number of pickups and that of drop-offs. That is, $\ell_i = \ell_i^p - \ell_i^d$, with $\ell_0^p = \ell_0^d = \ell_0 = 0$ and $\ell_{N+1}^p = \ell_{N+1}^d = \ell_{N+1} = 0$. Moreover, for each $k \in Z_1$, w_k^p and w_k^d denote the walking times of passenger k before being picked up and after being dropped off, respectively, and $c\ell_k^p$, $c\ell_k^d$ denote the indexes of the clusters that contain E_k^p and E_k^d , respectively. It should be noted that ℓ_i^p , ℓ_i^d , ℓ_i , $c\ell_k^p$, $c\ell_k^d$, $\forall i \in \{0\} \cup Z_2 \cup \{N+1\}, k \in Z_2$ are constants associated with the clustering pattern.

With the above definitions, the master problem in Equation (8) can be restated as an MIQCP formulation,

$$\text{Min } C = \gamma_1 T_{N+1} + \gamma_2 \sum_{k=1}^n \left(\alpha_1 T_{c\ell_k^p} + \alpha_2 (T_{c\ell_k^d} - T_{c\ell_k^p}) + \alpha_3^p w_k^p + \alpha_3^d w_k^d \right), \quad (9)$$

subject to

$$\sum_{j \in Z_2} x_{0j} = 1, \quad (10)$$

$$\sum_{i \in \{0\} \cup Z_2} x_{ij} = 1, \quad \forall j \in Z_2, \quad (11)$$

$$\sum_{j \in Z_2 \cup \{N+1\}} x_{ij} = 1, \quad \forall i \in Z_2, \quad (12)$$

$$\sum_{i \in Z_2} x_{i(N+1)} = 1, \quad (13)$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in Z_3, \quad (14)$$

$$T_j \geq (T_i + t_{ij} + t_a)x_{ij}, \quad \forall (i, j) \in Z_3, \quad (15)$$

$$T_{c\ell_k^d} \geq T_{c\ell_k^p} + t_{(c\ell_k^p)(c\ell_k^d)} + t_a, \quad \forall k \in Z_1, \quad (16)$$

$$c \geq q_i, \quad \forall i \in \{0\} \cup Z_2 \cup \{N+1\}, \quad (17)$$

$$q_j \geq (q_i + \ell_j)x_{ij}, \quad \forall (i, j) \in Z_3, \quad (18)$$

$$f_j^p \geq (f_i^p + \ell_i^p)x_{ij}, \quad \forall (i, j) \in Z_3, \quad (19)$$

$$f_j^d \geq (f_i^d + \ell_i^d)x_{ij}, \quad \forall (i, j) \in Z_3, \quad (20)$$

$$|(f_{c\ell_k^p}^p + 1) - k| \leq MPS^p, \quad \forall k \in Z_1, \quad (21)$$

$$|(f_{c\ell_k^d}^d + 1) - k| \leq MPS^d, \quad \forall k \in Z_1, \quad (22)$$

$$R_i \in A_i, \quad \forall i \in \{0\} \cup Z_2, \quad (23)$$

$$v_s^2 t_{ij}^2 = \|R_i - R_j\|_2^2, \quad \forall (i, j) \in Z_4, \quad (24)$$

$$t_{i(N+1)} = 0, \quad \forall i \in Z_2, \quad (25)$$

$$v_p^2 w_k^{p2} = \|P_k - R_{c\ell_k^p}\|_2^2, \quad \forall k \in Z_1, \quad (26)$$

$$v_p^2 w_k^{d2} = \|D_k - R_{c\ell_k^d}\|_2^2, \quad \forall k \in Z_1, \quad (27)$$

$$w_k^p \leq T_{c\ell_k^p}, \quad \forall k \in Z_1. \quad (28)$$

Equation (9) is a restatement of the objective function in the MIQCP formulation, which is equivalent to that in Equation (4). T_{N+1} corresponds to the departure time from cluster C_{N+1} , which equals the finishing time of the trip. $T_{c\ell_k^p}$ represents the difference between the pickup time and the request time, therefore, equals the waiting time of passenger k . Similarly, $T_{c\ell_k^d} - T_{c\ell_k^p}$ represents the difference between the drop-off time and the pickup time and, hence, equals the k th passenger's riding time.

Constraint Equations (10)–(13) ensure that the shuttle departs from the depot O , ends the trip at the dummy cluster C_{N+1} , and visits every other cluster exactly once. Consistence of the departure times is guaranteed by constraints Equation (15), which essentially serve as the subtour elimination conditions. Equations (16) and (17) ensure that the legitimate constraints and capacity constraints are satisfied, respectively. Equations (18)–(20) guarantee the consistence of the numbers of passengers on board, finished pickups and drop-offs at anytime of the trip, respectively. Equations (21) and (22) ensure that the MPS constraints are obeyed. Equations (23) essentially guarantee that each routing point is within the corresponding area. Equations (24) and (25) define the travel time of the shuttle between each pair of routing points, and Equations (26) and (27) defines the walking times

of the passengers before being picked up and after being dropped off, respectively. Equations (28) ensure that the shuttle departs from each area after all passengers whose pickups are associated have already arrived at the corresponding routing point and gotten on board.

This formulation is nonlinear because of constraint Equations (15), (18)–(20), (23), (24), (26), and (27). Of these, we note that Equations (15) and (18)–(20) are intrinsically linear and can be simplified into equivalent linear formats by introducing extra explaining constants. For example, the subtour elimination constraints in Equation (15) are equivalent to

$$T_j \geq T_i + t_{ij} + t_a - T_{ij}(1 - x_{ij}) \quad (29)$$

for each $(i, j) \in Z_3$, where $T_{ij} \geq T_i + t_{ij} + t_a$ and $T_j \geq 0$. Simply set $T_{ij} = T_c$, where $T_c \geq \max_{(i,j) \in Z_3} (T_i + t_{ij} + t_a)$ in the implementation. This technique is very similar to that used in the TSP formulation in References [11, 21]. Using the same procedure, Equations (18)–(20) can be linearized as follows:

$$q_j \geq q_i + \ell_j - Q_{ij}(1 - x_{ij}), \quad (30)$$

$$f_j^p \geq f_i^p + \ell_i^p - F_{ij}^p(1 - x_{ij}), \quad (31)$$

$$f_j^d \geq f_i^d + \ell_i^d - F_{ij}^d(1 - x_{ij}), \quad (32)$$

where $Q_{ij} \geq q_i + \ell_j$ and $q_j \geq 0$. Similarly, set $Q_{ij} = Q_c$, where $Q_c \geq \max_{(i,j) \in Z_3} (q_i + \ell_j)$. And $F_{ij}^p \geq f_i^p + \ell_i^p$, $F_{ij}^d \geq f_i^d + \ell_i^d$, $f_j^p, f_j^d \geq 0$. Set $F_{ij}^p = F_{ij}^d = F_c$, where $F_c \geq \max\{\max_{i \in Z_2} (f_i^p + \ell_i^p), \max_{i \in Z_2} (f_i^d + \ell_i^d)\}$.

The remaining constraint Equations (23), (24), (26), and (27) contain nonlinearities, which stem from the inherent quadratic nature of the Euclidean distance metric, which cannot be linearized unless via approximation. Among them, Equations (23) are convex as the space window for each pickup/drop-off event is a circle, which is convex, and therefore, the intersection of any combination of the original space windows is convex as well. Equations (24), (26), and (27) are non-convex due to the equality sense in the definitions of the time cost terms. Typical methods such as convex relaxation with Second Order Cone (SOC) constraints can be utilized to accommodate the non-convexity. One such approach is detailed below. For each $(i, j) \in Z_3$ and each $k \in Z_2$, introduce rotational explaining variables

$$R_{ij}^r = \frac{R_i - R_j}{v_s} \quad (33)$$

and shifted explaining variables

$$R_k^{s,p} = \frac{R_c \ell_k^p - P_k}{v_p}, \quad R_k^{s,d} = \frac{R_c \ell_k^d - D_k}{v_p}. \quad (34)$$

Therefore, the SOC formulations of Equations (24), (26), and (27) are

$$\|R_{ij}^r\|_2^2 \leq t_{ij}^2, \quad (35)$$

$$\|R_k^{s,p}\|_2^2 \leq w_k^{p^2}, \quad \|R_k^{s,d}\|_2^2 \leq w_k^{d^2}, \quad (36)$$

where $t_{ij} \geq 0$, and $w_k^p, w_k^d \geq 0$.

Such an SOC-based relaxation approach becomes exact during the process of the minimization of Equation (9), which guarantees that the SOC constraint Equations (35) and (36) are equivalent to the original definitions of the time cost terms in Equations (24), (26), and (27). The overall dimensionality of the decision variables and constraints of the MIQCP formulation are summarized in Table 1. We note that if Manhattan distance or metrics that correspond to real street topologies rather than a Euclidean distance metric is used, the problem can be further reduced into an MIP formulation with linear constraints.

Table 1. Dimensionality of the Decision Variables and Constraints of the MIQCP Formulation

	Category	Dimensionality
Variables	Binary	$N^2 + N$
	Continuous	$3N^2 + 7N + 6n + 10$
	Sum	$4N^2 + 8N + 6n + 10$
Constraints	Quadratic	$2n$
	SOC	$N^2 + 2n$
	Linear	$6N^2 + 7N + 10n + 11$
	Sum	$7N^2 + 7N + 14n + 11$

Given that the master problem in Equation (8) is now formulated to a standard MIQCP, commercial solvers, such as Gurobi in our implementation, can be used to find the optimal solution with sufficient computational time. This is because the number of feasible sequences is finite, and Gurobi basically traverses all possible sequence and computes the corresponding optimal routing points. However, the computational complexity is huge as both the dimensionality of the decision variable as well as constraints are quadratic in the size of the problem (n, N). For example, if 10 passengers are to be served and 16 clusters are formed, there are 1,222 decision variables in total, 272 of which are binary. The total number of constraints is 2,055 and 20 of them are quadratic, 276 are SOC and the rest are linear. It can be computationally highly burdensome, and necessitates the development of a few heuristics. This is the subject of Section 4.

4 AN ALTERNATING MINIMIZATION ALGORITHM

We propose an AltMin algorithm in this section to solve the master problem formulated in Equation (8). Due to its simplicity and effectiveness, AltMin has long been a popular optimization method, dating back to early works in the optimization literature (e.g., Reference [24]), and has been widely studied under various assumptions (e.g., References [8, 19]). It consists of a block coordinate descent approach, and operates by adjusting one block of coordinates at a time to attain successive reductions in the objective function. Given the presence of two sets of decision variables S and R in the master problem under consideration, and given that they are quite distinct from another, the use of such an approach is fairly natural. We therefore develop an AltMin algorithm with two phases, where in Phase 1, we optimize C over S , keeping R fixed, while in Phase 2, we keep S fixed and optimize C over R . A combination of consecutive Phase 1 plus Phase 2 constitutes one *mega iteration* in our AltMin algorithm.

More formally, let (S^*, R^*) denote the desired decision variables that optimizes C in Equation (8). That is

$$(S^*, R^*) = \underset{(S, R) \in S_f \times R_f}{\operatorname{argmin}} C(S, R). \quad (37)$$

It should be noted that (S^*, R^*) can be derived exactly via the MIQCP formulation in Section 3. The AltMin algorithm is proposed to solve (S^*, R^*) approximately via alternately iterating Phase 1 and Phase 2, where at the h th mega iteration, $\forall h \in \mathbb{Z}_{>0}$, Phases 1 and 2 correspond to updates

$$S^h = \underset{S \in S_f}{\operatorname{argmin}} C(S, R^{h-1}), \quad (38)$$

$$R^h = \underset{R \in R_f}{\operatorname{argmin}} C(S^h, R), \quad (39)$$

respectively, where R^0 is the initialization of the routing points. The mega iterations are terminated with a suitable stopping criterion.

The advantages of the AltMin algorithm are mainly that it decomposes the master problem, which is complicated and extremely computationally costly, into two subproblems, which are both fairly easy to solve. And in many circumstances, it is powerful in terms of optimality and computational efficiency [12, 23, 30]. Specifically, in our routing task, compared with the MIQCP formulation, which seeks to find the optimal solution by traversing all feasible sequences, the AltMin algorithm restricts its search to a subset of the entire solution space, which is determined judiciously through successive Phase 1-Phase 2 optimizations.

One possible drawback is that the decoupling process of S and R results in the intrinsic heuristic nature of AltMin. The convergence and optimality properties of the generic AltMin algorithm can be found in various references such as References [13, 15, 17, 29], which follow under certain regularity conditions. However, the mixed presence of both discrete and continuous decision variables introduces significant challenges in providing similar analytical guarantees for convergence and optimality. Instead, in this manuscript, we focus on a numerical demonstration of the superiority of AltMin over MIQCP using real operational data over a range of scenarios. Before proceeding to this demonstration in Section 5, we discuss details of Phases 1 and 2 of the proposed AltMin algorithm in this section.

To start the AltMin algorithm, the routing points need to be initialized as R^0 . For simplicity and consistency, each element R_i^0 of R^0 is computed as the center of mass of A_i , $\forall i \in \{0\} \cup Z_2$, in our implementation. With this initialization, we discuss details of Phase 1 in Section 4.1, details of Phase 2 in Section 4.2, convergence analysis and stopping criterion in Section 4.3.

4.1 Phase 1

As mentioned above, in Phase 1, the routing points R are fixed, and the focus is on the optimization of C over S . In our problem formulation, the shuttle is assumed to travel between the areas directly following the straight lines connecting the corresponding routing points; therefore, the statuses of the system of when the shuttle is at each cluster are sufficient to determine the sequence of visit. The status of the system when the shuttle is at each cluster is uniquely determined by: (1) the cluster it is current at and (2) the other clusters that have been visited. With this assumption, we define the state vectors of the system, and derive the underlying recurrence relation that corresponds to Phase 1 using these states. This is described in Sections 4.1.1–4.1.3.

4.1.1 State Vector. Let an $N + 1$ dimensional vector $(L, e_1, e_2, \dots, e_N)$ denote the state of the system where

- $L \in \{0\} \cup Z_2$ is the index of the cluster that the shuttle is currently at. $L = 0$ corresponds to the shuttle being at the depot, and $1 \leq L \leq N$ corresponds to the shuttle being at cluster C_L .
- $e_i \in \{0, 1\}$, $\forall i \in Z_2$ is a flag that indicates whether the shuttle has visited cluster C_i or not, with $e_i = 1$ if the shuttle has visited or is currently at cluster C_i , and $e_i = 0$, otherwise.

The definition of the state vector implies that the *initial state* of the shuttle could only be $(0, 0, 0, \dots, 0)$. And due to the open-ended setting of our formulation, the shuttle is eligible to finish the trip with any feasible cluster except for C_0 . Therefore, a *terminal state* is expressed as $(L, 1, 1, \dots, 1)$, $\forall L \in Z_2$.

4.1.2 Feasibility of State Vectors. A state vector is feasible if and only if: (1) it is consistent by definition, (2) it satisfies all primary constraints specified in Section 2.3. Each of these cases is detailed below.

4.2.2.1 Consistency. A state vector should be consistent by definition, essentially: if $L = 0$, it follows that $e_i = 0, \forall i \in Z_2$; otherwise if $1 \leq L \leq N$, it follows that $e_L = 1$.

4.2.2.2 Capacity Constraints. Recalling for each $i \in Z_2$, ℓ_i is the net load, which corresponds to the difference between the number of passengers been picked up and that of passengers been dropped off, at cluster C_i . It follows that capacity constraints are satisfied if

$$\sum_{i \mid e_i=1, i \in Z_2} \ell_i \leq c. \quad (40)$$

The left-hand side of Equation (40) computes the sum of net loads of the shuttle from departure until the current state, therefore, equates to the number of passengers who are currently on board.

4.2.2.3 MPS Constraints. $\sum_{i \mid e_i=1, i \in Z_2 \setminus \{L\}} \ell_i^p$ and $\sum_{i \mid e_i=1, i \in Z_2 \setminus \{L\}} \ell_i^d$ correspond to the number of total finished pickups and drop-offs prior to visiting the current cluster C_L , respectively. Therefore, the MPS constraints can be verified as follows:

$$\left| \left(\sum_{i \mid e_i=1, i \in Z_2 \setminus \{L\}} \ell_i^p + 1 \right) - k \right| \leq MPS^p, \quad \forall k \mid E_k^p \in C_L, \quad (41)$$

$$\left| \left(\sum_{i \mid e_i=1, i \in Z_2 \setminus \{L\}} \ell_i^d + 1 \right) - k \right| \leq MPS^d, \quad \forall k \mid E_k^d \in C_L. \quad (42)$$

4.2.2.4 Legitimate Constraints. The legitimate constraints are accommodated via the fact that a state is feasible only if there exist at least one feasible *next state*, as long as it is not a terminal state. A next state of a given state is defined as one that is eligible to occur immediately after the current state. A state vector $(L', e'_1, e'_2, \dots, e'_N), L' \neq 0$, where L' is the index of the *next cluster* of the current state, can be a next state if and only if

- (1) The cluster $C_{L'}$ has not been visited yet, i.e., $e_{L'} = 0$. This condition will automatically guarantee that all passengers to be picked up/dropped off have not been picked up/dropped off yet, i.e., the events contained in $C_{L'}$ have not yet occurred.
- (2) The passengers to be dropped off have already been picked up, i.e., $\forall E_{k_d}^d \in C_{L'}, e_{c \ell_{k_d}^p} = 1$.
- (3) The status of the clusters will remain the same except for that of cluster $C_{L'}$, i.e., $e'_{L'} = 1$ while $e'_i = e_i, \forall i \in Z_2$ and $i \neq L'$.

This screening of feasible next states guarantees that the legitimate constraints of all passengers are satisfied. The secondary constraints, which are defined in Definition 2.10, are not considered in Phase 1 as they do not affect the feasibility of the sequence of clusters to be visited.

4.1.3 Recurrence Relation. Once the feasibility verifications of all possible states have been conducted, the next step is optimality considerations, which is accomplished via a backward recursion. Let $V(L, e_1, e_2, \dots, e_N)$ be the optimal system cost measured in terms of C defined in Equation (4) of all subsequent decisions from the current state $(L, e_1, e_2, \dots, e_N)$ onwards till the end of the trip, i.e., any feasible terminal state is reached. It should be noted that V is only defined for the states that are feasible. It is clear that $C^* = V(0, 0, 0, \dots, 0)$ and $V = 0$ for any terminal state. The recurrence relation of $V(L, e_1, e_2, \dots, e_N)$ is, therefore, given by

$$V(L, e_1, e_2, \dots, e_N) = \begin{cases} \min_{\forall L'} \left[t(L, L') \cdot M + V(L', e'_1, e'_2, \dots, e'_N) \right] & \text{if not a terminal state} \\ 0 & \text{otherwise.} \end{cases} \quad (43)$$

The term $t(L, L') \cdot M$ denotes the subcomponent of C that corresponds to the incremental time cost incurred due to the travel of the shuttle from cluster C_L to $C_{L'}$. It is distributed over the first,

second, and third term in Equation (4), since it includes time costs incurred by the shuttle, the passengers waiting to be picked up, and those that are riding on the shuttle. Therefore, M is given by

$$M = \gamma_1 + \gamma_2 \left(\alpha_1 \sum_{i \mid e_i=0, i \in Z_2} \ell_i^p + \alpha_2 \sum_{i \mid e_i=1, i \in Z_2} \ell_i \right). \quad (44)$$

It should be noted that $t(L, L') = \frac{\|R_L - R_{L'}\|_2}{v_s} + t_a$ is the time taken for the shuttle to travel from C_L to $C_{L'}$. It is possible to calculate $t(L, L')$ off line as the routing points are fixed prior to this phase.

4.1.4 Remarks. Several remarks regarding Phase 1 are discussed as follows:

- (1) The sequence of the clusters to be visited consists of a series of sequential transitions of feasible system states. Let $\mathcal{E} = \sum_{i \in Z_2} e_i$ denote the *entropy* of the state $(L, e_1, e_2, \dots, e_N)$. One can see that $\mathcal{E} = 0$ in the initial state, increased by one at each state transition and ending with $\mathcal{E} = N$ at any feasible terminal state. In the implementation of Phase 1, the optimization relation between adjacent states in Equation (43) only occurs at states that are feasible. Equation (43) is implemented as a backward recursion, which starts from all feasible terminal states, decreases the entropy by one at each step, and finds the optimal next state for every feasible state, until the initial state is reached. The optimal next state for all feasible states is memorized in this manner. Together, the overall AltMin algorithm starts from the initial state, traces forward through successive optimal next states, and puts together the overall optimal route.
- (2) *Computational Complexity.* By definition, there are at most $(N + 1)2^N$ possible states to begin with, which are decreased to $1 + N2^{N-1}$ after a consistency check. For each of these states, the further feasibility check based on the primary constraints and the optimization screening on the therefore feasible states both take $O(n)$ time. Hence, the overall running time is $O(nN2^{N-1})$. Due to the presence of the primary constraints, the number of feasible states is in fact much smaller than $1 + N2^{N-1}$, though this is not able to improve the asymptotic running time of Phase 1, it helps accelerate the algorithm significantly in actual computational experiments. In particular, the algorithm adopts a Dynamic Programming (DP) paradigm, which provides an exact solution for Phase 1, though the computational complexity is exponential with respect to the problem size, it is asymptotically better than classical DP approaches [25].
- (3) It should be noted that the problem addressed here in Phase 1 is a generalized TSP. The specific approach used is akin to the Bellman-Held-Karp (BHK) algorithm [25]. Certain modifications have to be made, however, to address the fact that the algorithm has to derive the sequencing of sets of events, i.e. clusters, rather than sequencing single events. Moreover, the overall master problem is a even more generalized TSP, which is defined over areas, instead of points.
- (4) The modified BHK algorithm we have proposed is suitable for shuttles with capacity up to 15. If even larger capacities need to be considered, then one can either distribute the implementation of the modified BHK algorithm, e.g., system states with the same entropy have no dependency, therefore, their feasibility screenings and optimization considerations could both be parallelized to speed up the computation, or certain heuristics can be developed.
- (5) The main analytical property of the solution derived via the modified BHK algorithm to Phase 1 is summarized in Lemma 4.1.

LEMMA 4.1 (UNIQUENESS AND OPTIMALITY OF PHASE 1). $\forall h \in \mathbb{Z}_{\geq 0}$, given R^h , S^{h+1} is uniquely determined in (38) via Phase 1 and it is optimal.

PROOF. $\forall h \in \mathbb{Z}_{\geq 0}$, given R^h , S^{h+1} is derived via the modified BHK algorithm detailed in Section 4.1. The algorithm essentially examines all feasible states and seeks the sequence with minimal cost in a backward recursive manner; therefore, S^{h+1} is optimal and uniquely determined. $\square$

4.2 Phase 2

In Phase 2, the sequence of the clusters to be visited, S , is fixed, and the focus is on the optimization of C over the routing points, R , whose determination process can be formulated as a Quadratically Constrained Quadratic Programming (QCQP). The QCQP in Phase 2 is convex and can be solved to optimality via standard solvers such as Gurobi in our implementation. It should be noted that the primary constraints need not to be examined in this phase as the sequence is given while the secondary constraints are to be accommodated. The QCQP formulation is detailed as follows.

For each cluster $C_i, \forall i \in \{0\} \cup Z_2$, its routing point should be within the corresponding area, i.e., $R_i \in A_i$. And $t_i, w_k^p, w_k^d, \forall i \in Z_2, k \in Z_1$ are defined as in Section 3. Similarly, rotational and shifted explaining variables are introduced to transform the quadratic equality representations of the time cost terms to SOC constraints as follows:

$$R_i^r = \frac{R_{S(i-1)} - R_{S(i)}}{v_s}, \quad \forall i \in Z_2, \quad (45)$$

$$R_k^{s,p} = \frac{R_{c\ell_k^p} - P_k}{v_p}, \quad R_k^{s,d} = \frac{R_{c\ell_k^d} - D_k}{v_p}, \quad \forall k \in Z_1. \quad (46)$$

Therefore,

$$\|R_i^r\|_2^2 \leq t_i^2, \quad \forall i \in Z_2, \quad (47)$$

$$\|R_k^{s,p}\|_2^2 \leq w_k^{p^2}, \quad \|R_k^{s,d}\|_2^2 \leq w_k^{d^2}, \quad \forall k \in Z_1, \quad (48)$$

where $t_i \geq 0$ and $w_k^p, w_k^d \geq 0$. To satisfy the departure constraints, the walking time of any passenger to be picked up should not exceed the departure time of the shuttle from the cluster that contains the corresponding pickup event, which is rewritten as

$$w_k^p \leq \sum_{i=1}^{c\ell_k^p} t_i + S(c\ell_k^p) t_a, \quad \forall k \in Z_1. \quad (49)$$

Using the above notations, we rewrite Equation (4) as

$$C = \sum_{i=1}^N \left[\gamma_1 + \gamma_2 \left(\alpha_1 g_i^1 + \alpha_2 g_i^2 \right) \right] (t_i + t_a) + \gamma_2 \sum_{k=1}^n \left(\alpha_3^p w_k^p + \alpha_3^d w_k^d \right), \quad (50)$$

where

$$g_i^1 = n - \sum_{j=0}^{i-1} \ell_{S(j)}^p, \quad \forall i \in Z_2 \quad (51)$$

is the number of passengers who have not been picked up yet, therefore, are waiting when the shuttle is traveling on the i th trip segment, and

$$g_i^2 = \sum_{j=0}^{i-1} \ell_{S(j)}, \quad \forall i \in Z_2 \quad (52)$$

is the number of passengers who are on board when the shuttle is traveling on the i th trip segment.

Table 2. Dimensionality of the Decision Variables and Constraints of the QCQP Formulation for Phase 2

	Category	Dimensionality
Variables	Binary	-
	Continuous	$5N + 6n + 2$
	Sum	$5N + 6n + 2$
Constraints	Quadratic	$2n$
	SOC	$N + 2n$
	Linear	$2N + 5n + 2$
	Sum	$3N + 9n + 2$

With the above definitions, the underlying problem in Phase 2 is formulated as a QCQP, which minimizes C in Equation (50), subject to constraints defined in Equations (23) and (45)–(48). The uniqueness and optimality of the solution to the QCQP formulation is given by Lemma 4.2.

LEMMA 4.2 (UNIQUENESS AND OPTIMALITY OF PHASE 2). $\forall h \in \mathbb{Z}_{>0}$, given S^h, R^h is uniquely determined in Equation (39) via Phase 2 and it is optimal.

PROOF. $\forall h \in \mathbb{Z}_{>0}$, given S^h, R^h is derived by solving the QCQP formulation detailed in Section 4.2. $\forall i \in \{0\} \cup \mathbb{Z}_2$, A_i is strictly convex; therefore, R_f is strictly convex as well. For each time cost term in Equation (50), it is either an L_2 norm between two routing points (t_i) or an L_2 norm between one routing point and one requested location, which is constant (w_k^p or w_k^d), therefore, strictly convex as well. Thus a nonnegative weighted sum of the time cost terms, i.e., C , is strictly convex, as a result the QCQP formulation is strictly convex, thus R^h is optimal and uniquely determined. $\square$

The dimensionality of the decision variables and constraints of the QCQP formulation for Phase 2 of the AltMin algorithm are summarized in Table 2, where the number of variables and constraints are both linear with respect to the size of the problem, instead of quadratic dependency, due to the fact that the flow variables x_{ij} are no longer in need as the sequence is fixed. Moreover, all decision variables are continuous, which further makes the QCQP formulation easy and efficient to solve.

4.3 Convergence Analysis and Stopping Criterion

In this section, the convergence analysis of the AltMin algorithm is stated in Lemma 4.3. The stopping criterion of the mega iterations is discussed thereafter.

LEMMA 4.3 (CONVERGENCE OF THE ALTMIN ALGORITHM). *The AltMin algorithm either converges or recurs between a finite subset of the solution space.*

PROOF. S_f is a discrete set with finite cardinality, thus $\exists h_0 \in \mathbb{Z}_{>0}$, such that $S^{h_0} = S^{h'_0}$, for some $h'_0 \in \mathbb{Z}_{\geq 0}$ and $h'_0 < h_0$, and $\forall h \in \mathbb{Z}_{\geq 0}, h < h_0$, S^h are distinct. According to Lemma 4.1, $\forall h \in \mathbb{Z}_{>0}$, given R^{h-1} , S^h is uniquely determined via Phase 1. Similarly, according to Lemma 4.2, given S^h, R^h is uniquely determined via Phase 2. Therefore, $\forall h \in \mathbb{Z}_{\geq 0}, (S^{h_0+h}, R^{h_0+h}) = (S^{h'_0+h}, R^{h'_0+h})$, that being said, no new solution will be produced after mega iteration h_0 . Moreover, if $h_0 - h'_0 = 1$, (S^h, R^h) converges to $(S^{h'_0}, R^{h'_0})$; otherwise, (S^h, R^h) recurs between $(S^{h'_0}, R^{h'_0})$ and (S^{h_0-1}, R^{h_0-1}) . $\square$

It follows that in our implementation, once such an h_0 is encountered, the algorithm should stop as the subset of the solution space that can be reached by the AltMin algorithm has all been

ALGORITHM 1: The AltMin Algorithm of Static Routing with Space Windows

Input: depot O , ride requests $\{P_k, D_k, r_k^p, r_k^d, T_k^r\}, \forall k \in Z_1$, clustering pattern $\{C_1, C_2, \dots, C_N\}$, hyper parameters $\gamma_1, \gamma_2, \alpha_1, \alpha_2, \alpha_3^p, \alpha_3^d, c, MPS^p, MPS^d$, and mega iterations upper bound $h_{\max}$

Initialization: $C^* = \infty, S^* = \emptyset, R^* = \emptyset, R_i^0 \leftarrow$ center of mass of $A_i, \forall i \in \{0\} \cup Z_2$

```

for  $h = 1 : h_{\max}$  do
   $S^h \leftarrow \underset{S \in S_f}{\operatorname{argmin}} C(S, R^{h-1})$ 
  if  $\exists h' < h, h' \in \mathbb{Z}_{>0}, \text{ s.t. } S^h = S^{h'}$  then
    break
  end
   $R^h \leftarrow \underset{R \in R_f}{\operatorname{argmin}} C(S^h, R)$ 
   $C^h \leftarrow \min_{R \in R_f} C(S^h, R)$ 
  if  $C_h < C^*$  then
     $C^* \leftarrow C_h$ 
     $S^* \leftarrow S_h$ 
     $R^* \leftarrow R_h$ 
  end
end
Output:  $C^*, S^*, R^*$ 

```

explored. In addition, an integer upper bound $h_{\max}$ is also set on the number of mega iterations to limit the overall computational time. As a result, the actual number of mega iterations that the AltMin algorithm will perform is $\bar{h} = \min\{h_0, h_{\max}\}$. Hence, the output of the algorithm is $C^* = \min_{h \in \{1, 2, \dots, \bar{h}\}} C_h$, denote $h^* = \operatorname{argmin}_{h \in \{1, 2, \dots, \bar{h}\}} C_h$ and $S^* = S^{h^*}, R^* = R^{h^*}$.

The overall pseudocode of the AltMin algorithm is summarized in Algorithm 1.

5 COMPUTATIONAL EXPERIMENTS

The AltMin algorithm described in Section 4 and the MIQCP formulation described in Section 3 are now numerically evaluated using real operational data. The data was collected from the shuttle operation conducted in October 2016 by Chariot [3, 16] in San Francisco. The current operation therein consists of receiving passengers' requests regarding pickup, drop-off locations and start times of the trips, based on which a static route is designed. To evaluate our proposed AltMin algorithm, these pickup and drop-off locations, i.e., longitude and latitude, are utilized, thereby providing a realistic demand pattern for the evaluation.

A total number of 26 instances with various scenarios, including size of the problem (n, N) , clustering patterns, passenger preferences, priority guarantees, and system objectives, have been simulated. For each of the instances, both the Altmin algorithm and the MIQCP-based algorithm are implemented in MATLAB R2016b, where QCQP and MIQCP are implemented in Gurobi 7.5.1 with a MATLAB interface. A 3.20 GHz Intel(R) Xeon(R) CPU E5-1660 v4 desktop with 128 GB of memory is used for all computational experiments. The fixed parameters are chosen such that $v_s = 30$ mph, $v_p = 3.1$ mph, $t_0 = 1$ min and $a = 2.25$ mph/s, which reasonably reflect the actual traffic conditions in urban areas.

The weights in Equation (4) are chosen as $\gamma_1 = 1$, and $\alpha_1 = 2, \alpha_2 = 1$, with the assumption that a large waiting time is less preferred than a large riding time by the passengers. The shuttle capacity is set as $c = 6$. A computational upper bound of one hour is set for all experiments. $h_{\max}$ in the

Table 3. Settings of the Computational Experiments

Index	Instance	Problem Size		Hyper Parameters			
		n	N	γ_2	α_3^p/α_3^d	r^p/r^d	MPS^p/MPS^d
1	p06-c12-1	6	12	0	-	0.3	6
2	p06-c06-1	6	6	0	-	0.3	6
3	p06-c12-2	6	12	1	0.1	0.3	6
4	p06-c07-1	6	7	1	0.1	0.3	6
5	p06-c12-3	6	12	1	0.1	0.3	2
6	p06-c07-2	6	7	1	0.1	0.3	2
7	p06-c12-4	6	12	0	-	0.15	6
8	p06-c08-1	6	8	0	-	0.15	6
9	p08-c16-1	8	16	0	-	0.3	6
10	p08-c08-1	8	8	0	-	0.3	6
11	p08-c16-2	8	16	1	0.1	0.3	6
12	p08-c09-1	8	9	1	0.1	0.3	6
13	p08-c16-3	8	16	1	0.1	0.3	2
14	p08-c11-1	8	11	1	0.1	0.3	2
15	p08-c16-4	8	16	0	-	0.15	6
16	p08-c09-2	8	9	0	-	0.15	6
17	p10-c20-1	10	20	1	0	0.3	6
18	p10-c20-2	10	20	0	-	0.3	6
19	p10-c12-1	10	12	0	-	0.3	6
20	p10-c20-3	10	20	1	0.1	0.3	6
21	p10-c11-1	10	11	1	0.1	0.3	6
22	p10-c20-4	10	20	1	1	0.3	6
23	p10-c20-5	10	20	1	0.1	0.3	2
24	p10-c13-1	10	13	1	0.1	0.3	2
25	p10-c20-6	10	20	0	-	0.15	6
26	p10-c16-1	10	16	0	-	0.15	6

AltMin algorithm is set to 50 as an early stopping condition. Other problem settings that are subject to change across different instances are specified in Table 3. Each of the 26 instances is designed to represent different application scenarios. For instance, all scenarios where $\gamma_2 = 0$ correspond to the cases where the objective of the routing task is only to minimize the total travel time of the shuttle. Scenarios where $MPS^p = MPS^d = 2$ correspond to the cases where more emphasis is placed on the queue order and the dynamic shuttle services are more in a first come first serve fashion.

The computational results obtained for the 26 instances are summarized in Table 4. Three columns are included to describe the performances of AltMin, and five for MIQCP. The three columns of AltMin correspond to $\bar{h}$, the actual number of overall iterations of Phase 1 and 2 when convergence is reached, C_{AltMin}^* (unit [h]) the cost of the system obtained, and the CPU time (unit [s]). And the five columns of MIQCP correspond to *Vars* and *Cons*, the total numbers of decision variables and constraints in the formulation, respectively, *Bound* (unit [h]) the best lower bound of the cost obtained in the branch-and-cut implementation of Gurobi, C_{MIQCP}^* cost of the system obtained, and the CPU time. In the latter, it follows that if the best lower bound equals C_{MIQCP}^* , then optimality is achieved. We include an additional column corresponding to *Optimality Gap* to

Table 4. Results of the Computational Experiments

Instance	AltMin			MIQCP					Optimality Gap
	h	C_{AltMin}^*	CPU	Vars	Cons	Bound	C_{MIQCP}^*	CPU	
◇p06-c12-1	2	0.469	2.409	706	1,187	0.356	0.469	3,600	-
*p06-c06-1	2	0.361	0.740	232	389	0.361	0.361	0.283	0
p06-c12-2	2	3.497	2.383	706	1,187	2.024	3.482	3,500	0.41%
p06-c07-1	2	2.914	1.009	291	487	2.886	2.886	0.343	0.96%
*p06-c12-3	2	3.482	1.856	706	1,187	3.482	3.482	716.2	0
*p06-c07-2	2	3.262	0.803	291	487	3.262	3.262	0.261	0
*p06-c12-4	2	0.504	2.355	706	1,187	0.504	0.504	2,368	0
*p06-c08-1	2	0.430	1.028	358	599	0.430	0.430	1.053	0
◇p08-c16-1	3	0.583	52.01	1,194	2,027	0.047	0.626	3,600	-
*p08-c08-1	2	0.548	4.645	370	627	0.548	0.548	1.889	0
◇p08-c16-2	2	5.439	34.13	1,194	2,027	2.052	5.446	3,600	-
p08-c09-1	2	4.926	6.228	445	753	4.538	4.438	5.188	8.55%
◇p08-c16-3	2	6.586	20.26	1,194	2,027	2.259	6.603	3,600	-
*p08-c11-1	2	5.995	6.689	619	1,047	5.995	5.995	8.172	0
◇p08-c16-4	2	0.629	33.76	1,194	2,027	0.052	0.670	3,600	-
*p08-c09-2	2	0.521	6.560	445	753	0.521	0.521	6.146	0
◇p10-c20-1	2	8.482	560.8	1,810	3,091	2.454	8.484	3,600	-
◇p10-c20-2	2	0.767	579.9	1,810	3,091	0.064	1.090	3,600	-
p10-c12-1	2	0.710	120.8	730	1,243	0.699	0.699	238.6	1.54%
◇p10-c20-3	2	8.658	551.4	1,810	3,091	2.567	9.227	3,600	-
*p10-c11-1	2	8.344	67.66	631	1,075	8.344	8.344	159.8	0
◇p10-c20-4	2	9.373	554.8	1,810	3,091	2.628	9.491	3,600	-
◇p10-c20-5	2	9.205	273.9	1,810	3,091	2.502	9.278	3,600	-
*p10-c13-1	3	7.895	87.83	837	1,425	7.895	7.895	65.74	0
◇p10-c20-6	3	0.817	752.2	1,810	3,091	0.052	1.072	3,600	-
◇p10-c16-1	3	0.769	401.8	1,206	2,055	0.066	0.848	3,600	-

benchmark the optimality of AltMin with respect to MIQCP, defined as

$$\text{optimality gap} = \frac{C_{\text{AltMin}}^* - C_{\text{MIQCP}}^*}{C_{\text{MIQCP}}^*}, \quad \text{if } C_{\text{MIQCP}}^* \leq C_{\text{AltMin}}^*. \quad (53)$$

If $C_{\text{MIQCP}}^* > C_{\text{AltMin}}^*$, then it is clear that AltMin outperforms MIQCP in terms of the cost optimality. A diamond symbol ◇ is included at the beginning of a row where Altmin outperforms MIQCP in terms of both optimality and computational efficiency. An asterisk symbol * is included at the beginning of the row where the optimality gap of the AltMin algorithm is zero.

It is clear from Table 4 that AltMin outperforms MIQCP in most of the instances, regarding both optimality and computational efficiency. More detailed observations are made below.

- AltMin is significantly faster than MIQCP. The average CPU time of all instances for AltMin is 158.8 seconds, which is more than one order smaller than that of 1,937 seconds for MIQCP. It should be noted that a one-hour computational cap is placed.

Fig. 5. Overall routes of four instances of the same set of requests with different problem settings.

- Among the 26 instances, AltMin converges to optimality in 10 instances, and with better computational efficiency than MIQCP; average CPU time for AltMin is 19.81 seconds versus 290.1 seconds for MIQCP. These instances are the ones marked by an asterisk * in Table 4.
- In 12 out of the 26 instances marked by a diamond $\diamond$, AltMin outperforms MIQCP in terms of both optimality and computational efficiency.
- In only 4 instances, MIQCP yields a better cost than AltMin; however, the optimality gap is fairly small. This indicates that in some scenarios, even though AltMin converges to a suboptimal solution, it is fairly close to the optimal. The computational time remains much smaller in these cases as well when compared to MIQCP.
- In all of the computational experiments, AltMin displays a fast convergence, with only 2 or 3 overall iterations of Phase 1 and 2.

For different instances with the (n, N) , a smaller r^p/r^d or MPS^p/MPS^d yields faster computations and better optimality, for both AltMin and MIQCP. This is due to the fact that though the problem sizes are the same, smaller r^p/r^d or MPS^p/MPS^d values shrink the size the solution space (though the asymptotic running time still remain the same), which facilitate faster and more optimal solutions. Similarly, a smaller c value shares same impacts, which is not demonstrated here.

The computational efficiency of the AltMin algorithm can be improved even further by (i) utilizing more powerful computational resources and (ii) exploiting heuristics for both Phase 1 (e.g., k -opt, simulated annealing, ant colony optimization) and Phase 2 (e.g., greedy algorithms).

Figure 5 demonstrates the overall routes of four instances derived using AltMin. It should be noted that the set of requests, i.e., requested pickup and drop-off locations, are the same for the

four instances. The reason why the actual routes realized are different is due to the differences in the corresponding problem settings shown in Table 3.

6 EXTENSION TO DYNAMIC ROUTING

In this section, the algorithm for dynamic routing is discussed using static routing as a subcomponent. Dynamic routing with space window preserves the advantages of its static counterpart, and is more flexible in terms of how to accommodate new requests and achieve better system level performance. In the dynamic routing scheme, the shuttle server searches for new requests periodically and makes suitable updates (using a modified version of the proposed static routing algorithm, detailed below) on its current route to take care of the new requests immediately as long as all constraints can be satisfied and a reduction on the cost of the system can be achieved.

The main difference between dynamic and static routing is that any update on the route can affect those passengers who have either been assigned pickups but have not been picked up yet or already been on the shuttle, such that their actual pickups/drop-offs might be delayed. We denote these passengers as *existing passengers*. However, as long as the delays are below certain upper bounds specified in the initial ride offers provide by the shuttle server, the delays are considered acceptable.

Overall, one can view dynamic routing as a generalized version of its static counterpart, with a periodical update on the static route in response to real time requests from new passengers. Below, we summarize how we modify the AltMin algorithm described in Section 4 and the MIQCP formulation described in Section 3 to accommodate this update. It should be noted that the same assumption is made that the updated clustering pattern is explicitly given beforehand.

(1) Modified dummy initial cluster C_0 .

In static routing, the dummy initial cluster is set as $C_0 = \emptyset$. In dynamic routing, C_0 needs to be modified to accommodate the passengers who are already on board when updating the route. In particular, C_0 consists of the pickup events of the passengers who are on board (if any), with the corresponding A_0 as the shuttle's exact location when the update occurs.

(2) Frozen routing points of existing but unexecuted pickup events.

The clustering pattern, sequence of the clusters to be visited and routing points are all subject to change when any update occurs. An additional constraint in dynamic routing should be included that the routing points corresponding to the pickup events of the existing passengers who are yet to be picked up have to be frozen, and are no longer decision variables. Otherwise, additional walking times may be imposed on these passengers. Though the overall walking distance might still be within the specified maximum walking distance r_k^p , the *double walk* may cause huge negative externalities to the riding experiences of these passengers, which is not desirable. This can be easily accomplished by setting $S_W(P_k, r_k^p) = S_W(R_{cl_k^p}, 0), \forall k$ that corresponds to an existing passenger who has not yet been picked up. It should be noted that the negotiated drop-off locations of the existing passengers are not necessarily frozen as the walks to the final destinations have not yet occurred, thus no such double walk will happen because of the updates on the current route (Figure 6).

(3) Consistent references of time frame and queues.

Static routing only takes care of one batch of requests, while dynamic routing conducts static routing periodically, where consecutive batches of requests are accommodated. The references of time frame and the three queues associated with the passengers, i.e., $\{k\}$, $\{q_k\}$ and $\{d_k\}$ should be consistent throughout different batches. The consistency can be

Fig. 6. Demonstrations of double walks. (a) Double walks caused by updating R_1 , which serves as the negotiated pickup locations for passenger 1 and 2, who have not been picked up yet. (b) Double walks are prevented by freezing R_1 . (c) R_3 is flexible as it serves as the negotiated drop-off location for passenger 1 and 2, whose updates will not result in any double walk.

guaranteed by simply using the absolute references for both the time frame and the three queues. More specifically, for any passenger k , $\forall k \in \mathbb{Z}_{>0}$, k , p_k , and d_k are his/her absolute sequences in the entire request, pickup and drop-off queue, respectively, instead of the relative sequences in the current batch. And then the MPS constraints are able to guarantee that no passenger is likely to experience any indefinite deferment due to the updates on the routes. Similarly, the waiting and riding time of any passenger should be derived via collapsing between the absolute time stamps when he/she gets picked up and requests the shuttle ride, and between those when he/she gets dropped off and picked up, respectively.

Dynamic routing has been demonstrated to be advantageous over its static counterpart when the requests arrive frequently, more specifically, when the time intervals between different batches of requests are shorter than those it takes to serve all existing passengers [6]. However, static routing is easier to operate and understand, passengers are not subject to any delays on their pickups or drop-offs, and the drivers will not suffer from fatigue due to frequent updates on the shuttle routes (certainly this issue could be alleviated significantly if autonomous driving vehicles could come into play). Therefore, if the requests are not frequent compared to the duration of the existing trips, then static routing suffices. However, in any scenario with a high travel demand, dynamic routing may be unavoidable and necessary.

Figure 7 demonstrates a numerical example of how the AltMin dynamic routing algorithm operates to serve a total number of 12 passengers whose requests are received in 2 batches, 6 requests per batch. The location data (i.e., longitude and latitude of requested pickup and drop-off locations) is based on the *GoRide* dynamic shuttle service on Ford's Dearborn campus in August 2016 [2]. The first batch is supposed to be finished in 24.3 minutes (derived via the AltMin static routing algorithm), while the second batch was received at $T^r = 15$ minutes, when passenger 2 and 4 are on aboard and passenger 3 has not been picked up yet. Figures 7(d) and 7(e) compare the overall routes to serve all 12 passengers using static and dynamic routing, respectively. It is clear that via dynamic routing, the shuttle makes suitable updates, e.g., R_7 , and achieves a better system level performance.

7 SUMMARY AND CONCLUSIONS

In this article, we have proposed a novel dynamic routing framework for shared mobility services that utilizes multi-passenger shuttles, facilitated by the central concept of space window. In addition to the development of the framework, the contributions of the article include a constrained

Fig. 7. Demonstrations of dynamic routing via AltMin. (a, b) The Origin-Destination (OD) pairs of the two batches of requests, respectively. (c) The original route of the first batch, dashed arrows correspond to the part that have not been carried out yet when the second batch is received; (d, e) the overall routes to serve all 12 passengers, via static and dynamic routing, respectively. One tick corresponds to 0.5 mile.

optimization formulation that shows how the space window concept can be utilized to carry out dynamic routing of shuttles, and an AltMin algorithm composed of a modified BHK algorithm as Phase 1 and a QCQP formulation as Phase 2 to solve the constrained optimization problem. As shown in Section 5, this AltMin algorithm is demonstrated to be an order of magnitude better in terms of computational efficiency and comparable in terms of optimality to a standard MIQCP-based approach. Overall, the proposed dynamic routing framework is capable of reducing the underlying travel time costs, and enriches the scope of the emerging shared MoD services, providing an ideal combination of flexibility, convenience, and affordability for people to move around.

Future works will concentrate on the development of heuristics to solve Phase 1 of the AltMin algorithm to accommodate the scenarios where larger capacity shuttles are in use. An analytical and more comprehensive analysis on the optimality of the AltMin algorithm, as well as approaches such as simulated annealing so that the overall iterations of Phases 1 and 2 do not end in a local minimum, are of interest. The determination process of the clustering pattern will also be investigated systematically. The extension to a distributed optimization framework for a multi-shuttle scenario and that to dynamic pricing strategies [6] are our future directions as well.

REFERENCES

- [1] ReinventingParking.org. [n.d.]. “Cars are parked 95% of the time.” Let’s check! Retrieved from <http://www.reinventingparking.org/2013/02/cars-are-parked-95-of-time-lets-check.html>.
- [2] MediaFord.com. [n.d.]. Ford Dynamic Shuttle service moves from experiment to pilot, providing point-to-point shuttle rides for Employess. Retrieved from <https://media.ford.com/content/fordmedia/fna/us/en/news/2015/12/10/ford-dynamic-shuttle-service-moves-from-experiment-to-pilot.html>.

- [3] MediaFord.com. [n.d.]. Ford partnering with global cities on new transportation; Chariot Shuttle to be acquired, Ford Gobike to launch in San Francisco. Retrieved from <https://media.ford.com/content/fordmedia/fna/us/en/news/2016/09/09/ford-partnering-with-global-cities-on-new-transportation--chario.html>.
- [4] Javier Alonso-Mora, Samitha Samaranyake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* 114, 3 (2017), 462–467.
- [5] Giorgio Ambrosino, J. D. Nelson, and Maurizio Romanazzo. 2004. *Demand Responsive Transport Services: Towards the Flexible Mobility Agency*. ENEA, Italian National Agency for New Technologies, Energy and the Environment.
- [6] Anuradha M. Annaswamy, Yue Guan, H. Eric Tseng, Hao Zhou, Thao Phan, and Diana Yanakiev. 2018. Transactive control in smart cities. *Proc. IEEE* 106, 4 (2018), 518–537.
- [7] Bilge Atasoy, Takuro Ikeda, Xiang Song, and Moshe E. Ben-Akiva. 2015. The concept and impact analysis of a flexible mobility on demand system. *Transport. Res. Part C: Emerg. Technol.* 56 (2015), 373–392.
- [8] Alfred Auslender. 1976. Optimisation. *Méthodes numériques*.
- [9] Timo Berthold, Stefan Heinz, and Stefan Vigerske. 2012. Extending a CIP framework to solve MIQCP s. In *Mixed Integer Nonlinear Programming*. Springer, 427–444.
- [10] Z. J. Chong, Baoxing Qin, Tirthankar Bandyopadhyay, Tichakorn Wongpiromsarn, Brice Rebsamen, P. Dai, E. S. Rankin, and Marcelo H. Ang Jr. 2013. Autonomy for mobility on demand. In *Intelligent Autonomous Systems 12*. Springer, 671–682.
- [11] Jean-François Cordeau. 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operat. Res.* 54, 3 (2006), 573–586.
- [12] Imre Csiszár. 1984. Information geometry and alternating minimization procedures. *Statistics and Decisions* 1 (1984), 205–237.
- [13] Luigi Grippo and Marco Sciandrone. 2000. On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. *Operat. Res. Lett.* 26, 3 (2000), 127–136.
- [14] IBM. 2016. Building a smarter transportation management network. Retrieved from <https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=LBW03019USEN>.
- [15] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*. ACM, 665–674.
- [16] Paul Kampe. [n.d.]. Ford-Owned Ride-Sharing Service Chariot Expands to New York City. Retrieved from <https://www.at.ford.com/en/homepage/news-and-clipsheet/news/2017/7/chariot0.html>.
- [17] Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Comput.* 19, 10 (2007), 2756–2779.
- [18] Jennie Eugénie Lioris and Guy Cohen. 2016. Optimised flexible transport mode without advanced reservations by a DES approach. *IFAC-PapersOnLine* 49, 18 (2016), 974–979.
- [19] Zhi-Quan Luo and Paul Tseng. 1993. Error bounds and convergence analysis of feasible descent methods: A general approach. *Ann. Operat. Res.* 46, 1 (1993), 157–178.
- [20] Elliot W. Martin and Susan A. Shaheen. 2011. Greenhouse gas emission impacts of carsharing in North America. *IEEE Trans. Intell. Transport. Syst.* 12, 4 (2011), 1074–1086.
- [21] Clair E. Miller, Albert W. Tucker, and Richard A. Zemlin. 1960. Integer programming formulation of traveling salesman problems. *J. ACM* 7, 4 (1960), 326–329.
- [22] William J. Mitchell et al. 2008. Mobility on demand: Future of transportation in cities. *Smart Cities MIT Media Laboratory, Technical Report*.
- [23] Urs Niesen, Devavrat Shah, and Gregory W. Wornell. 2009. Adaptive alternating minimization algorithms. *IEEE Trans. Info. Theory* 55, 3 (2009), 1423–1429.
- [24] James M. Ortega and Werner C. Rheinboldt. 2000. *Iterative Solution of Nonlinear Equations in Several Variables*. SIAM.
- [25] Harilaos N. Psaraftis. 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transport. Sci.* 14, 2 (1980), 130–154.
- [26] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H. Strogatz, and Carlo Ratti. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proc. Natl. Acad. Sci. U.S.A.* 111, 37 (2014), 13290–13294.
- [27] Kevin Spieser, Kyle Treleven, Rick Zhang, Emilio Frazzoli, Daniel Morton, and Marco Pavone. 2014. Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in Singapore. In *Road Vehicle Automation*. Springer, 229–245.
- [28] Remi Tachet, Oleguer Sagarra, Paolo Santi, Giovanni Resta, Michael Szell, S. H. Strogatz, and Carlo Ratti. 2017. Scaling law of urban ride sharing. *Sci. Rep.* 7 (2017).
- [29] Paul Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.* 109, 3 (2001), 475–494.

- [30] Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang. 2008. A new alternating minimization algorithm for total variation image reconstruction. *SIAM J. Imag. Sci.* 1, 3 (2008), 248–272.
- [31] Hyejin Youn, Michael T. Gastner, and Hawoong Jeong. 2008. Price of anarchy in transportation networks: Efficiency and optimality control. *Phys. Rev. Lett.* 101, 12 (2008), 128701.
- [32] Biying Yu, Ye Ma, Meimei Xue, Baojun Tang, Bin Wang, Jinyue Yan, and Yi-Ming Wei. 2017. Environmental benefits from ridesharing: A case of Beijing. *Appl. Energy* 191 (2017), 141–152.
- [33] Desheng Zhang, Juanjuan Zhao, Fan Zhang, Ruobing Jiang, and Tian He. 2015. Feeder: Supporting last-mile transit with extreme-scale urban infrastructure data. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. ACM, 226–237.

Received December 2017; revised September 2018; accepted December 2018