Optimization of Ride Sharing Systems Using Event-driven Receding Horizon Control ⋆

Rui Chen * **Christos G. Cassandras** *

* *Division of Systems Engineering and Center for Information and Systems Engineering, Boston University, Brookline, MA, USA, (e-mail: ruic@bu.edu, cgc@bu.edu).*

Abstract: We develop an event-driven Receding Horizon Control (RHC) scheme for a Ride Sharing System (RSS) where vehicles are shared to pick up and drop off passengers so as to minimize a weighted sum of passenger waiting and traveling times. The RSS is modeled as a discrete event system and the event-driven nature of the controller significantly reduces the complexity of the vehicle assignment problem, thus enabling its real-time implementation. Simulation results using actual city maps and real traffic data illustrate the effectiveness of the RH controller in terms of online implementation and performance relative to known heuristics.

Keywords: Transportation system; Ride sharing system; Event-driven receding horizon control; Applications; Discrete approaches for hybrid systems; Performance evaluation, optimization;

1. INTRODUCTION

It has been abundantly documented that the state of transportation systems worldwide is at a critical level. Disruptive technologies that aim at dramatically altering the transportation landscape include vehicle connectivity and automation as well as shared personalized transportation through emerging mobility-on-demand systems. Focusing on the latter, the main idea of a Ride Sharing System (RSS) is to assign vehicles in a given fleet so as to serve multiple passengers, thus effectively reducing the total number of vehicles on a road network, hence also congestion, energy consumption, and adverse environmental effects.

The main objectives of a RSS are to minimize the total Vehicle-Miles-Traveled (VMT) over a given time period (equivalently, minimize total travel costs), to minimize the average waiting and traveling times experienced by passengers, and to maximize the number of satisfied RSS participants (both drivers and passengers) (Agatz et al. [2012]). When efficiently managed, a RSS has the potential to reduce the total number of private vehicles in a transportation network, hence also decreasing overall energy consumption and traffic congestion. From a passenger standpoint, a RSS is able to offer door-to-door transportation with minimal delays which makes traveling more convenient. From an operator's standpoint a RSS provides a considerable revenue stream. A RSS also provides an alternative to public transportation or can work in conjunction with it to reduce possible low utilization of vehicles and long passenger delays.

In this paper, we concentrate on designing dynamic vehicle assignment strategies in a RSS aiming to minimize the system-wide waiting and traveling times of passengers. The main challenge in obtaining optimal vehicle assignments is the complexity of the optimization problem involved in conjunction with uncertainties such as random passenger service request times, origins, and destinations, as well as unpredictable traffic conditions which determine the times to pick up and drop off passengers. Algorithms used in RSS are limited by the NP-complete nature of the underlying traveling salesman problem (Chen et al. [2017]) which is a special case of the much more complex problems encountered in RSS optimization. Therefore, a global optimal solution for such problems is generally intractable, even in the absence of the aforementioned uncertainties. Moreover, a critical requirement in such algorithms is a guarantee that they can be implemented in a real-time context.

Several methods have been proposed to solve the RSS problem addressing the waiting and traveling times of passengers (Agatz et al. [2011], Santi et al. [2014], Berbeglia et al. [2010], Berbeglia et al. [2010], Alonso-Mora et al. [2017], Chen et al. [2017], Calafiore et al. [2017], Tsao et al. [2018], Salazar et al. [2018]). In this paper, we adopt an *event-driven Receding Horizon Control* (RHC) approach. The basic idea of event-driven RHC introduced in Li and Cassandras [2006] and extended in Khazaeni and Cassandras [2018] is to solve an optimization problem over a given *planning horizon* when an event is observed in a way which allows vehicles to cooperate; the resulting control is then executed over a generally shorter *action horizon* defined by the occurrence of the next event of interest to the controller. Compared to methods such as Santi et al. [2014] and Alonso-Mora et al. [2017], the RHC scheme is not constrained by vehicle seating capacities and is specifically designed to dynamically re-allocate waiting passengers to vehicles at any time. Moreover, compared to the time-driven strategy in Chen et al. [2017], the event-driven RHC scheme refrains from unnecessary calculations when no event in the RSS occurs. Finally, in contrast to models used in Tsao et al. [2018] and Salazar et al. [2018], we maintain control of every vehicle and passenger in a RSS at a microscopic level while ensuring that real-time optimal (over each receding horizon) vehicle assignments can be made.

In Section 2, we present a discrete event system model of a RSS and formulate an optimization problem aimed at minimizing a weighted sum of passenger waiting and traveling times. Section 3 reviews the basic RHC scheme

⋆ Supported in part by NSF under grants ECCS-1931600, DMS-1664644, CNS-1645681, by AFOSR under grant FA9550-19-1-0158, by ARPA-E's NEXTCAR program under grant DE-AR0000796 and by the MathWorks.

previously used and identifies how it is limited in the context of a RSS. This motivates the new RHC approach described in Section 4, specifically designed for a RSS. Simulation results are given in Section 5 based on real traffic data and we conclude with Section 6.

2. PROBLEM FORMULATION

We consider a Ride Sharing System (RSS) in a traffic network consisting of N nodes $\mathcal{N} = \{1, ..., N\}$ where each node corresponds to an intersection. Nodes are connected by arcs (i.e., road segments). Thus, we view the traffic network as a directed graph $\mathbb{G}$ which is embedded in a two-dimensional Euclidean space and includes all points contained in every arc, i.e., $\mathbb{G} \subset \mathbb{R}^2$. In this model, a node $n \in \mathcal{N}$ is associated with a point $\nu_n \in \mathbb{G}$, the actual location of this intersection in the underlying two-dimensional space. The set of vehicles present in the RSS at time t is $\mathcal{A}(t)$, where the index $j \in \mathcal{A}(t)$ will be used to uniquely denote a vehicle, and let $A(t) = |\mathcal{A}(t)|$. The set of passengers is $\mathcal{P}(t)$, where the index i will be used to uniquely denote a passenger, and let $P(t) = |\mathcal{P}(t)|$. Note that $\mathcal{A}(t)$ and $\mathcal{P}(t)$ are time-varying.

There are two points in $\mathbb{G}$ associated with each passenger i, denoted by $o_i, r_i \in \mathbb{G}$: o_i is the origin where the passenger issues a service request (pickup point) and r_i is the passenger's destination (drop-off point). Let $O(t) = \{o_1, ..., o_P\}$ be the set of all passenger origins and $R(t) = \{r_1, ..., r_P\}$ the corresponding destination set. Vehicles pick up passengers and deliver them to their destinations according to some policy. We assume that the times when vehicles join the RSS are not known in advance, but they become known as a vehicle joins the system. Similarly, the times when passenger service requests occur are random and their destinations become known only upon being picked up.

State Space: In addition to $\mathcal{A}(t)$ and $\mathcal{P}(t)$ describing the state of the RSS, let $x_j(t) \in \mathbb{G}$ and $N_j(t) \in \{0, 1, ..., C_j\}$ be the position and number of passengers of vehicle j at time t where C_j is the capacity of j. The state of passenger i is denoted by $s_i(t)$ where $s_i(t) = 0$ if passenger i is waiting to be picked up and $s_i(t) = j \in \mathcal{A}(t)$, where $j > 0$, when the passenger is in vehicle j after being picked up. Finally, we associate with passenger i a left-continuous clock value $z_i(t) \in \mathbb{R}$ whose dynamics are defined as follows: when the passenger joins the system and is added to $\mathcal{P}(t)$, the initial value of $z_i(t)$ is 0 and we set $\dot{z}_i(t) = 1$. Thus, $z_i(t)$ may be used to measure the waiting time of passenger i. When i is picked up by some vehicle j at time $\rho_{i,j}$, $z_i(t)$ is reset to zero and thereafter measures the traveling time until the passenger's destination is reached at time $\sigma_{i,j}$. In summary, the state of the RSS is $\mathbf{X}(t) = \{\mathcal{A}(t), x_1(t), \ldots, x_A(t), N_1(t), \ldots, N_A(t), \mathcal{P}(t), s_1(t), \ldots, s_P(t), z_1(t), \ldots, z_P(t)\}$.

Events: All state transitions in the RSS are event-driven with the exception of the passenger clock states $z_i(t)$, $i \in \mathcal{P}(t)$, in which case it is the reset conditions that are event-driven. As we will see, all control actions (to be defined) affecting the state $\mathbf{X}(t)$ are taken only when an event takes place. Therefore, regarding a vehicle location $x_j(t)$, $j \in \mathcal{A}(t)$, for control purposes we are interested in its value only when events occur, even though we assume that $x_j(t)$ is available to the RSS for all t based on an underlying localization system.

We define next the set E of all events whose occurrence causes a state transition. We set $E = E_U \cup E_C$ to differentiate between uncontrollable events in E_U and controllable events in E_C. There are six possible event types: (1) $\alpha_i \in E_U$: a service request is issued by passenger i. (2) $\beta_j \in E_U$: vehicle j joins the RSS. (3) $\gamma_j \in E_U$: vehicle j leaves the RSS. (4) $\pi_{i,j} \in E_C$: vehicle j picks up passenger i (at $o_i \in \mathbb{G}$). (5) $\delta_{i,j} \in E_C$: vehicle j drops off passenger i (at $r_i \in \mathbb{G}$). (6) $\zeta_{m,j} \in E_C$: vehicle j arrives at intersection (node) $m \in \mathcal{N}$.

Note that events α_i, β_j are uncontrollable exogenous events. Event γ_j is also uncontrollable, however it may not occur unless the "guard condition" $N_j(t) = 0$ is satisfied, i.e., vehicle j is empty when it leaves the system. The remaining three events are controllable. First, $\pi_{i,j}$ depends on the control policy (to be defined) through which a vehicle is assigned to a passenger and is feasible only when $s_i(t) = 0$ and $N_j(t) < C_j$. Second, $\delta_{i,j}$ is feasible only when $s_i(t) = j \in \mathcal{A}(t)$. Finally, $\zeta_{m,j}$ depends on the policy (to be defined) and occurs when the route taken by vehicle j involves intersection $m \in \mathcal{N}$.

State Dynamics: The events defined above determine the state dynamics as follows.

(1) Event α_i adds an element to the passenger set $\mathcal{P}(t)$ and increases its cardinality, i.e., $P(t^+) = P(t) + 1$ where t is the occurrence time of this event. In addition, it initializes the passenger state and associated clock: $s_i(t^+) = 0$, $\dot{z}_i(t^+) = 1$ with $z_i(t) = 0$ and generates the origin information $o_i \in \mathbb{G}$.

(2) Event β_j adds an element to the vehicle set $\mathcal{A}(t)$ and increases its cardinality, i.e., $A(t^+) = A(t) + 1$. It also initializes $x_j(t)$ to the location of vehicle j at time t.

(3) Event γ_j removes vehicle j from $\mathcal{A}(t)$ and decreases its cardinality, i.e., $A(t^+) = A(t) - 1$.

(4) Event $\pi_{i,j}$ occurs when $x_j(t) = o_i$ and it generates the destination of this passenger $r_i \in \mathbb{G}$. This event affects the states of both vehicle j and passenger i: $N_j(t^+) = N_j(t) + 1$, $s_i(t^+) = j$ and, since the passenger was just picked up, the associated clock is reset to 0 and starts measuring traveling time towards the destination r_i: $z_i(t^+) = 0$, $\dot{z}_i(t^+) = 1$.

(5) Event $\delta_{i,j}$ occurs when $x_j(t) = r_i$ and it causes a removal of passenger i from $\mathcal{P}(t)$ and decreases its cardinality, i.e., $P(t^+) = P(t) - 1$. In addition, it affects the state of vehicle j: $N_j(t^+) = N_j(t) - 1$.

(6) Event $\zeta_{m,j}$ occurs when $x_j(t) = \nu_m$. This event triggers a potential change in the control associated with vehicle j as described next.

Control: The control we exert is denoted by $u_j(t) \in \mathbb{G}$ and sets the destination of vehicle j in the RSS. We note that the destination $u_j(t)$ may change while vehicle j is en route to it based on new information received as various events may take place. The control is initialized when event β_j occurs at some point $x_j(t)$ by setting $u_j(t) = \nu_m$ where $m \in \mathcal{N}$ is the intersection closest to $x_j(t)$ in the direction vehicle j is headed. Subsequently, the vector $\mathbf{u}(t) = \{u_1(t), \ldots, u_A(t)\}$ is updated according to a given policy whenever an event from the set E occurs (we assume that all events are observable by the RSS controller). Our control policy is designed to optimize the objective function described next.

Objective Function: Our objective is to minimize the combined *waiting* and *traveling* times of passengers in the RSS over a given finite time interval $[0, T]$. In order

to incorporate all passengers who have received service over $[0,T]$, we define $\mathcal{P}_T = \cup_{t\in[0,T]}\mathcal{P}(t)$ to include all $i \in \mathcal{P}(t)$ for any $t \in [0,T]$. In simple terms, $\mathcal{P}_T$ records all passengers who are either currently active in the RSS at $t = T$ or were active and departed at some $t < T$ when the associated $\delta_{i,j}$ event occurred for some $j \in \mathcal{A}(t)$.

We define w_i to be the waiting time of passenger i and note that $w_i = z_i(t)$ where t is the time when event $\pi_{i,j}$ occurs. Similarly, letting y_i be the total traveling time of passenger i, we have $y_i = z_i(t)$ where t is the time when event $\delta_{i,j}$ occurs. We then formulate the following problem, given an initial state $\mathbf{X}_0$ of the RSS:

$$\min_{\mathbf{u}(t)} E\left[\sum_{i\in\mathcal{P}_T}[\mu_w w_i + \mu_y y_i]\right] \quad (1)$$

where μ_w, μ_y are weight coefficients defined so that $\mu_w = \frac{\omega}{W_{\max}}$ and $\mu_y = \frac{1-\omega}{Y_{\max}}$, $\omega \in [0,1]$, and $W_{\max}$ and $Y_{\max}$ are upper bounds of the waiting and traveling time of passengers respectively. The values of $W_{\max}$ and $Y_{\max}$ are selected to capture the worst case tolerated for waiting and traveling times respectively. This construction ensures that w_i and y_i are properly normalized so that (1) is well-defined.

The expectation in (1) is taken over all random event times in the RSS defined in an appropriate underlying probability space. Clearly, modeling the random event processes so as to analytically evaluate this expectation is a difficult task. This motivates viewing the RSS as evolving over time and adopting a control policy based on *observed* actual events and on *estimated* future events that affect the RSS state. Assuming for the moment that the system is deterministic, let t_k denote the occurrence time of the kth event over $[0,T]$. A control action $\mathbf{u}(t_k)$ may be taken at t_k and, for simplicity, is henceforth denoted by $\mathbf{u}_k$. Along the same lines, we denote the state $\mathbf{X}(t_k)$ by $\mathbf{X}_k$. Letting K_T be the number of events observed over $[0,T]$, we have the optimal value of the objective function when the initial state is $\mathbf{X}_0$. Next, we convert this into a maximization problem by considering $[-\mu_w w_i - \mu_y y_i]$ for each $i \in \mathcal{P}_T$. Moreover, observing that both w_i and y_i are upper-bounded by T, we consider the non-negative rewards $T - w_i$ and $T - w_i$ and rewrite $J(\mathbf{X}_0)$ as

$$J(\mathbf{X}_0) = \max_{u_0,\cdots u_{K_T}}\left[\sum_{i\in\mathcal{P}_T}[\mu_w(T-w_i) + \mu_y(T-y_i)]\right] \quad (2)$$

Then, determining an optimal policy amounts to solving the following Dynamic Programming (DP) equation:

$$J(\mathbf{X}_k) = \max_{\mathbf{u}_k\in\mathbb{G}}[C(\mathbf{X}_k,\mathbf{u}_k) + J_{k+1}(\mathbf{X}_{k+1})], \quad k = 0,1,\ldots,K_T$$

where $C(\mathbf{X}_k,\mathbf{u}_k)$ is the immediate reward at state $\mathbf{X}_k$ when control $\mathbf{u}_k$ is applied and $J_{k+1}(\mathbf{X}_{k+1})$ is the future reward at the next state $\mathbf{X}_{k+1}$. Our ability to solve this equation is limited by the well-known "curse of dimensionality" even if our assumption that the RSS is fully deterministic were to be valid. This further motivates adopting a *Receding Horizon Control* (RHC) approach as in similar problems encountered in Li and Cassandras [2006] and Khazaeni and Cassandras [2018]. This is in the same spirit as Model Predictive Control (MPC) techniques (Camacho and Alba [2013]) with the added feature of exploiting the event-driven nature of the control process, hence avoiding unnecessary calculations and can significantly improve the efficiency of the RH controller by reacting to random events as they occur in real time. In particular, a control action taken when the kth event is observed is selected to maximize an immediate reward defined over a *planning horizon* H_k, denoted by $C(\mathbf{X}_k,\mathbf{u}_k,H_k)$, followed by an estimated future reward $\hat{J}_{k+1}(\mathbf{X}(t_k+H_k))$ when the state is $\mathbf{X}(t_k+H_k)$. The optimal control action $\mathbf{u}_k^*$ is, therefore,

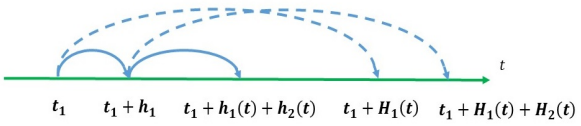

Fig. 1. Event-Driven receding horizon control.

$$\mathbf{u}_k^* = \arg\max_{\mathbf{u}_k\in\mathbb{G}}[C(\mathbf{X}_k,\mathbf{u}_k,H_k) + \hat{J}_{k+1}(\mathbf{X}(t_k+H_k))] \quad (3)$$

The control action $\mathbf{u}_k^*$ is subsequently executed only over a generally shorter *action horizon* $h_k \le H_k$ so that $t_{k+1} = t_k + h_k$ (see Fig. 1). The selection of H_k and h_k will be discussed in the next section.

3. RECEDING HORIZON CONTROL (RHC)

The basic RHC scheme in Li and Cassandras [2006] considers a set of cooperating "agents" and a set of "targets" in a Euclidean space. The purpose of agents is to visit targets and collect a certain time-varying reward associated with each target. The key steps of the scheme are as follows: (1) Determine a planning horizon H_k at the current time t_k. (2) Solve an optimization problem to minimize an objective function defined over the time interval $[t_k, t_k+H_k]$. (3) Determine an action horizon h_k and execute the optimal solution over $[t_k, t_k+h_k]$. (4) Set $t_{k+1} = t_k + h_k$ and return to step (1).

Letting $\mathcal{A}(t)$ be the agent set and $\mathcal{P}(t)$ the target set, we define $d_{i,j}(t)$ for any $i \in \mathcal{P}(t)$, $j \in \mathcal{A}(t)$ to be the Euclidean distance between target i and agent j at time t. In Li and Cassandras [2006], the planning horizon H_k is defined as the earliest time that any agent can visit any target. The action horizon h_k is defined to be the earliest time in $[t_k, t_k+H_k]$ when an event in the system occurs (e.g., a new target appears) or through $h_k = \epsilon H_k$ for some $\epsilon \in (0,1]$ so as to ensure that $h_k \le H_k$.

The concept of *neighborhood* for a target is defined in Li and Cassandras [2006] to allow the RHC to avoid early commitments of agents to target visits, since changes in the system state may provide a better opportunity for an agent to improve the overall system performance.

Using a relative responsibility function, the optimization problem solved by the RHC at each control action point assigns an agent to a point which minimizes a given objective function and which is not necessarily a target point. Details of how this problem is set up and solved and the properties of the original RHC scheme may be found in Li and Cassandras [2006]. In what follows, we define the distance between any two points $x, y \in \mathbb{R}^2$ expressed as $d(x,y)$

Limitations of the original RHC scheme: There are three main limitations of the original RHC scheme: *(1) Agent trajectory instabilities* may arise. *(2) Future cost estimation inaccuracies*: this future cost is estimated through its lower bound, thus resulting in an overly "optimistic" outlook. *(3) Algorithm complexity:* the optimization problem at each algorithm iteration involves the selection of each agent's heading over $[0,2\pi]$ as defined by the set of feasible reachable points $F_j(t_k,H_k) = \{w : d(w,x_j(t_k)) = vH_k\}$. The complexity of this problem is $O(G^{A(t)})$ where G is a given discretization level over $[0,2\pi]$.

The modified RCH scheme in Khazaeni and Cassandras [2018] deals with issues *(1)* and *(3)* above by defining a set of *active targets* $S_j(t_k, H_k)$ for agent j at each iteration time t_k. Its purpose is to limit the feasible reachable set $F_j(t_k, H_k)$ defined by all agent headings over $[0, 2\pi]$ so that it is reduced to a finite set of points. We omit details here which may be found in Chen and Cassandras [2019].

4. THE NEW RHC SCHEME

Although Khazaeni and Cassandras [2018] resolves some of the limitations of the original RHC scheme, it is still not suitable for a RSS, starting with the fact that the latter operates in a graph, rather than Euclidean, topology. Thus, we begin by introducing some variables used in the new RHC scheme as follows.

(1) $d(u, v)$ is defined as the *Manhattan distance* (Farris [1972]) between two points $u, v \in \mathbb{G}$. This measures the shortest path distance between two points on a directed graph that includes points on an arc of this graph which belong to $\mathbb{G} \subset \mathbb{R}^2$.

(2) $\mathcal{R}_{i,j}(t)$ is the set of the n *closest pickup locations* in the sense of the Manhattan distance defined above, where $n = C_j - N_j(t) - 1$ if j picks up i at o_i at time t, and $n = C_j - N_j(t) + 1$ if j drops off i at r_i at time t. Clearly, the set may contain fewer than n elements if there are insufficient pickup locations in the RSS at time t.

(3) $\hat{\mathcal{R}}_{i,j}(t)$ is the set of n *drop-off locations* for j, where $n = N_j(t) + 1$ if j picks up i at o_i, and $n = N_j(t) - 1$ if j drops off i at r_i.

(4) φ_i and $\rho_{i,j}$ denote the occurrence time of events α_i (passenger i joins the RSS) and $\pi_{i,j}$ (pickup of passenger i by vehicle j) respectively.

In the rest of this section we present the new RHC scheme which overcomes the issues previously discussed through four modifications: (i) We define the *travel value* of a passenger to a vehicle considering the distance between vehicles and passengers, as well as the vehicle's residual capacity. (ii) Based on the new travel value and the graph topology, we introduce a new *active target set* for each vehicle during $[t_k, t_k + H_k)$. This allows us to reduce the feasible solution set of the optimization problem (3) at each iteration. (iii) We develop an improved future reward estimation mechanism to better predict the time that a passenger is served in the future. (iv) To address the potential instability problem, a method to restrain oscillations is introduced in the optimization algorithm at each iteration.

We begin by defining the planning horizon H_k at the kth control update consistent as

$$H_k = \min_{i \in \mathcal{P}(t_k), j \in \mathcal{A}(t_k)} \left\{ \frac{d(x_j(t_k), c_i)}{v_j(t_k)} \right\} \tag{4}$$

where

$$c_i = \begin{cases} o_i & \text{if } s_i(t) = 0 \text{ and } N_j(t_k) < C_j \\ r_i & \text{if } s_i(t) = j \end{cases} \tag{5}$$

and $v_j(t_k)$ is the maximal speed of vehicle j at time t_k, assumed to be maintained over $[t_k, t_k + H_k]$. Thus, H_k is the shortest Manhattan distance from any vehicle location to any target (either o_i or r_i) at time t_k. Note that c_i is undefined if $s_i(t) = 0$ and $N_j(t_k) = C_j$. Formally, to ensure consistency, we set $d(x_j(t_k), c_i) = \infty$ if $s_i(t) = 0$ and $N_j(t_k) = C_j$ since o_i is not a valid pickup point for j in this case.

The action horizon $h_k \leq H_k$ is defined by the occurrence of the next event in E, i.e., $h_k = \tau_{k+1} - t_k$ where τ_{k+1} is the time of the next event to occur after t_k. If no such event occurs over $[t_k, t_k + H_k]$, we set $h_k = H_k$.

4.1 Vehicle Travel Value Function

In Khazaeni and Cassandras [2018], a travel cost function $\eta_i(x, t)$ was defined for any agent measuring the cost of traveling from a point x at time t to a target $i \in \mathcal{P}(t)$. In our case, we define instead a *travel value* measuring the reward (rather than cost) associated with a vehicle j when it considers any passenger $i \in \mathcal{P}(t)$. There are three cases to consider depending on the state $s_i(t)$ for any $i \in \mathcal{P}(t)$ as follows:

$$V_{i,j}(x_j(t), t) = \begin{cases} (1-\mu) \cdot \dfrac{t - \varphi_i}{W_{\max}} \\ +\mu \cdot \dfrac{D - d(x_j(t), o_i)}{D} & \text{if } s_i(t) = 0 \\ (1-\mu) \cdot \dfrac{t - \rho_{i,j}}{Y_{\max}} \\ +\mu \cdot \dfrac{D - d(x_j(t), r_i)}{D} & \text{if } s_i(t) = j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Case 1: If $s_i(t) = 0$, then passenger i is waiting to be picked up. From a vehicle j's point of view, there are two components to the value of picking up this passenger at point o_i: (i) The accumulated waiting time $t - \varphi_i$ of passenger i; the larger this waiting time, the higher the value of this passenger is. (ii) The distance of j from o_i; the shorter the distance, the higher the value of this passenger is. To ensure this value component is non-negative, we define D to be the largest possible travel time between any two points in the RSS (often referred to as the diameter of the underlying graph) and consider $D - d(x_j(t), o_i)$ as this value component. In order to properly normalize each component and ensure its associated value is restricted to the interval $[0, 1]$, we use the waiting time upper bound $W_{\max}$ introduced in (1) and the distance upper bound D. Finally, $\mu \in [0, 1]$ is a weight coefficient depending on the relative importance the RSS places on passenger satisfaction (measured by waiting time) and vehicle distance traveled.

Case 2: If $s_i(t) = j \in \mathcal{A}(t)$, then passenger i is already on board with destination r_i. We use similar idea as in *Case 1* to define the value.

In addition to this "immediate" value associated with passenger i, there is a future value for vehicle j to consider depending on the sets $\mathcal{R}_{i,j}(t)$ and $\hat{\mathcal{R}}_{i,j}(t)$ defined earlier. In particular, if $s_i(t) = 0$ and vehicle j proceeds to the pickup location o_i, then the value associated with $\mathcal{R}_{i,j}(t)$ is defined as $V^{\mathcal{R}}_{i,j}(x_j(t), t) = \max_{n \in \mathcal{R}_{i,j}(t)} V_{n,j}(o_i, t)$ which is the maximal travel value among all passengers in $\mathcal{R}_{i,j}(t)$ to be collected if vehicle j selects o_i as its destination at time t. On the other hand, if $s_i(t) = j$ and vehicle j proceeds to the drop-off location r_i, then $V_{n,j}(o_i, t)$ above is replaced by $V_{n,j}(r_i, t)$. Since the value of $s_i(t)$ is known to j, we will use c_i as defined in (5) and write $V^{\mathcal{R}}_{i,j}(x_j(t), t) = \max_{n \in \mathcal{R}_{i,j}(t)} V_{n,j}(c_i, t)$. Similarly, the value of $\hat{\mathcal{R}}_{i,j}(t)$ is defined as $V^{\hat{\mathcal{R}}}_{i,j}(x_j(t), t) = \max_{n \in \hat{\mathcal{R}}_{i,j}(t)} V_{n,j}(c_i, t)$. We then define the total travel value associated with a vehicle j when it considers any passenger $i \in \mathcal{P}(t)$ as

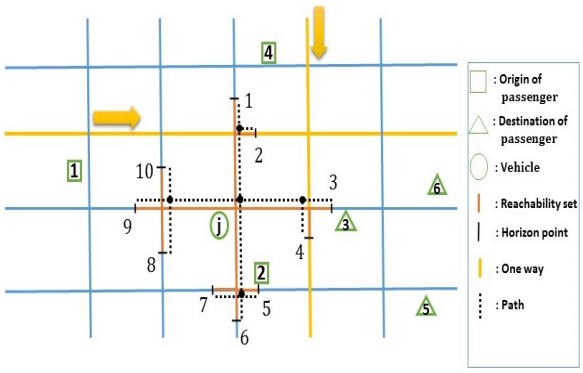

Fig. 2. Example of the reachability set of vehicle j.

$$\bar{V}_{i,j}(x_j(t),t) = V_{i,j}(x_j(t),t) + \max\{V^{\mathcal{R}}_{i,j}(x_j(t),t), V^{\hat{\mathcal{R}}}_{i,j}(x_j(t),t)\} \tag{7}$$

4.2 Active Target Sets

The active target set defined in Khazaeni and Cassandras [2018] cannot be used in a RSS since the topology is no longer Euclidean and the travel cost function $\eta_i(x,t)$ has been replaced by the travel value function (7). We begin by defining the reachability (or feasible) set $F_j(t_k, H_k)$ for vehicle j in the RSS topology specified by $\mathbb{G} \subset \mathbb{R}^2$. This is now a finite set consisting of *horizon points* in $\mathbb{G}$ reachable through some path starting from $x_j(t_k)$ and assuming a fixed speed $v_j(t_k)$ as defined in (4). We can now define the active target set of vehicle j to consist of any target (pickup or drop-off locations of passengers) which has the largest travel value to j for at least one horizon point $x \in F_j(t_k, H_k)$.

Definition: The set of *Active Targets* of vehicle j is defined as

$$S_j(t_k, H_k) = \{l : l = \arg\max_{i \in \mathcal{P}(t)} \bar{V}_{i,j}(x, t_k + H_k) \text{ for some } x \in F_j(t_k, H_k)\} \tag{8}$$

Observe that $S_j(t_k, H_k) \subseteq \mathcal{P}(t_k)$ and may reduce the number of passengers to consider as potential destinations assigned to j when $S_j(t_k, H_k) \subset \mathcal{P}(t_k)$ since $u_j(t_k) \in S_j(t_k, H_k)$. In the example of Fig. 2, $\mathcal{P}(t_k)$ contains 6 passengers where $s_1(t_k) = s_2(t_k) = s_4(t_k) = 0$ and $s_3(t_k) = s_5(t_k) = s_6(t_k) = j$. Thus, we can immediately see that $P(t_k) = 6 < |F_j(t_k, H_k)| = 10$. Further, observe that $r_5, r_6 \notin S_j(t_k, H_k)$ since both points are farther away from $x_j(t_k)$ than r_3 and o_2 respectively. Therefore, the optimal control selection to be considered at t_k is reduced to $u_j(t_k) \in S_j(t_k, H_k) = \{o_1, o_2, r_3, o_4\}$. In addition, if the capacity C_j happens to be such that $C_j = 3$, then the only feasible control would be $u_j(t_k) = r_3$.

4.3 Future Reward Estimation

In order to solve the optimization problem (3) at each RHC iteration time t_k, we need to estimate the time that a future target is visited when $t > t_k + H_k$ so as to evaluate the term $\hat{J}_{k+1}(\mathbf{X}(t_k + H_k))$. Let us start by specifying the immediate reward term $C(\mathbf{X}_k, \mathbf{u}_k, H_k)$ in (3). In view of (2), there are three cases: (i) As a result of $\mathbf{u}_k$, an event $\pi_{i,j}$ (where $s_i(t) = j$) occurs at time t_{k+1} with an associated reward $C(\mathbf{X}_k, \mathbf{u}_k, H_k) = \mu_w(T - w_i)$ where $w_i = t_{k+1} - \varphi_i$, (ii) As a result of $\mathbf{u}_k$, an event $\delta_{i,j}$ occurs at time t_{k+1} with an associated reward $C(\mathbf{X}_k, \mathbf{u}_k, H_k) = \mu_y(T - y_i)$ where $y_i = t_{k+1} - \rho_{i,j}$, and (iii) Any other event results in no immediate reward. In summary, adopting the notation $C(\mathbf{u}_k, t_{k+1})$ for the immediate reward resulting from control $\mathbf{u}_k$ (instead of $C(\mathbf{X}_k, \mathbf{u}_k, H_k)$ in (3)),we have

$$C(\mathbf{u}_k, t_{k+1}) = \begin{cases} \mu_w(T - w_i) & \text{if event } \pi_{i,j} \text{ occurs at } t_{k+1} \\ \mu_y(T - y_i) & \text{if event } \delta_{i,j} \text{ occurs at } t_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In order to estimate future rewards at times $t > t_{k+1}$, recall that $\mathcal{T}_{k,j} \subseteq \mathcal{P}(t) - \{u_j(t_k)\}$ is a set of targets that vehicle j would visit in the future, after reaching $u_j(t_k)$. This set was defined in Khazaeni and Cassandras [2018] and a new definition suitable for the RSS will be given below. Then, for each target $n \in \mathcal{T}_{k,j}$ the associated reward is $C(\mathbf{u}_k, \hat{\tau}_{n,j})$ where $\hat{\tau}_{n,j}$ is the estimated time that vehicle j reaches target n. If $n = o_i$ for some passenger i, then, from (9), $C(\mathbf{u}_k, \hat{\tau}_{n,j}) = \mu_w(T - \hat{w}_i)$ where $\hat{w}_i = \hat{\tau}_{n,j} - \varphi_i$, whereas if $n = r_i$ for some passenger i, then $C(\mathbf{u}_k, \hat{\tau}_{n,j}) = \mu_y(T - \hat{y}_i)$ where $\hat{y}_i = \hat{\tau}_{n,j} - \rho_{i,j}$. Further, we include a *discount factor* $\lambda_n(\hat{\tau}_{n,j})$ to account for the fact that the accuracy of our estimate $\hat{\tau}_{n,j}$ is monotonically decreasing with time, hence $\lambda_n(\hat{\tau}_{n,j}) \in (0, 1]$. Therefore, for each vehicle j the associated term for $\hat{J}_{k+1}(\mathbf{X}(t_k + H_k))$ is

$$\hat{J}_j(\mathbf{X}(t_k + H_k)) = \sum_{n=1}^{|\mathcal{T}_{k,j}|} \lambda_n(\hat{\tau}_{n,j}) C(u_{k,j}, \hat{\tau}_{n,j}) \tag{10}$$

We now need to derive estimates $\hat{\tau}_{n,j}$ for each $n \in \mathcal{T}_{k,j}$ and to define the set $\mathcal{T}_{k,j}$. We omit details here which may be found in Chen and Cassandras [2019].

4.4 Preventing Vehicle Trajectory Instabilities

This problem arises when a new passenger joins the system and introduces a new target for one or more vehicles in its vicinity which may have higher travel value in the sense of (7) than current ones. As a result, a vehicle may switch its current destination $u_j(t_k)$ and this process may repeat itself with additional future new passengers. In order to avoid frequent such switches, we introduce a threshold parameter denoted by Θ and react to any event α_i (a service request issued by a new passenger i). In simple terms, the current control remains unaffected unless the new passenger provides an incremental value relative to this control which exceeds a given threshold.

4.5 RHC optimization scheme

The full RHC scheme detailed in Chen and Cassandras [2019] consists of a sequence of optimization problems solved at each event time t_k, $k = 1, 2, \ldots$with each problem of the form

$$\mathbf{u}^*_k = \arg \max_{\substack{u_{k,j} \in S_j(t_k, H_k) \\ j \in \mathcal{A}(t_k)}} [C(\mathbf{u}_k, t_{k+1}) + \sum_{j \in \mathcal{A}(t_k)} \sum_{n=1}^{|\mathcal{T}_{k,j}|} \lambda_n(\hat{\tau}_{n,j}) C(u_{k,j}, \hat{\tau}_{n,j})] \tag{11}$$

where $S_j(t_k, H_k)$ is the active target of vehicle j at time t_k obtained through (8) and $C(\mathbf{u}_k, t_{k+1})$ is given by (9).

Complexity of Algorithm: As seen in (11), the optimal control for vehicle j at any iteration is selected from the finite active target set $S_j(t_k, H_k)$. Thus, the complexity is

Table 1. Average RHC performance for different weights ω (New York City RSS, 28 vehicles)

$[\omega, 1-\omega]$	Waiting time [mins]	Traveling time [mins]	Vehicle occupancy
$[0.05, 0.95]$	4.1	8.1	2.07
$[0.5, 0.5]$	5.2	12.4	2.79
$[0.95, 0.05]$	7.0	12.6	2.83

$O(\sum_{j\in\mathcal{A}(t)} S_j(t_k, H_k))$. Observe that $S_j(t_k, H_k)$ decreases as $H(k)$ is reduced and as targets are visited if new ones are not generated.

5. SIMULATION RESULTS

We use the SUMO (Simulation of Urban Mobility) (Behrisch et al. [2011]) transportation system simulator to evaluate our RHC for a RSS applied to two traffic networks (in Ann Arbor, MI and in New York City, NY). Vehicle speeds are set by the simulation and they include random factors like different road speed limits, turns, traffic lights, etc. The simulation results pertaining to the Ann Arbor RSS are omitted here and may be found in Chen and Cassandras [2019], as is the map covering an area of 10×10 blocks in New York City for which we generate passenger arrivals based on actual data from the NYC Taxi and Limousine Commission; these provide exact timing of arrivals and associated origins and destinations. We pre-load in SUMO a fixed number of vehicles and the remaining RSS system parameters are selected as follows: $C_j = 4$, $T = 300$ min, $W_{max} = 47$ min, $Y_{max} = 47$ min, $D = 3000$ m and the threshold Θ is set as 0.3. We pre-loaded 28 vehicles and run simulations until 160 passengers are served (which is within $T = 300$ min set above) based on actual data from a weekday of January, 2016 (the approximate passenger rate is 16 passengers/min). In order to evaluate the performance of the RSS at steady state, we allow a simulation to load more than 160 passengers.

Table 1 shows the associated waiting and traveling times under different weights. As expected, emphasizing waiting results in larger vehicle occupancy and longer average travel times. The associated waiting and traveling time histograms for all cases in Table 1 are shown in Chen and Cassandras [2019].

Table 2. Average RHC performance for different numbers of vehicles (New York City RSS, 160 passengers, $\omega = 0.5$)

Vehicle Numbers	Waiting time [mins]	Traveling time [mins]	Vehicle occupancy	Weighted sum in (1)
28	5.2	12.4	2.79	0.187
38	3.5	10.7	2.31	0.151

Table 2 compares different vehicle numbers when the delivered passenger number is 160 showing waiting and traveling times, vehicle occupancy and the objective in (1).

Finally, we tested a relatively longer RSS operation with 38 vehicles based on the same actual passenger data as before which generates 1000 passengers over approximately 1.2 real operation hours. Simulations will not end until 900 passengers are delivered. In Table 3, we compare our RHC method with a greedy heuristic (GH) algorithm (similar to Agatz et al. [2011]). As seen in Table 3 with $\omega = 0.5$, the RHC algorithm achieves a substantially better weighted sum performance (approximately by a factor of 2).

Table 3. Average waiting and traveling time comparisons under RHC and GH (New York City RSS, 38 vehicles, $\omega = 0.5$)

Method	Waiting time [mins]	Traveling time [mins]	Weighted sum in (1)
RHC	19.1	13.7	0.349
GH	61.4	19.0	0.855

6. CONCLUSIONS AND FUTURE WORK

An event-driven RHC scheme is developed for a RSS modeled as a discrete event system where vehicles are shared to pick up and drop off passengers so as to minimize a weighted sum of passenger waiting and traveling times. This event-driven nature reduces the complexity of the vehicle assignment problem. In our ongoing work, we are considering the load balancing problem for idle vehicles (Swaszek and Cassandras [2019]), including where to optimally position them so that they are best used upon receiving future calls. We will also study the case where destination information is known in advance.

REFERENCES

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

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

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., and Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3), 462–467.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo–simulation of urban mobility. In *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, volume 42.

Berbeglia, G., Cordeau, J.F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European journal of operational research*, 202(1), 8–15.

Calafiore, G.C., Novara, C., Portigliotti, F., and Rizzo, A. (2017). A flow optimization approach for the rebalancing of mobility on demand systems. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, 5684–5689. IEEE.

Camacho, E.F. and Alba, C.B. (2013). *Model predictive control*. Springer Science & Business Media.

Chen, R. and Cassandras, C.G. (2019). Optimization of ride sharing systems using event-driven receding horizon control. *arXiv preprint arXiv:1901.01919v2*.

Chen, X., Miao, F., Pappas, G.J., and Preciado, V. (2017). Hierarchical data-driven vehicle dispatch and ride-sharing. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, 4458–4463. IEEE.

Farris, J.S. (1972). Estimating phylogenetic trees from distance matrices. *The American Naturalist*, 106(951), 645–668.

Khazaeni, Y. and Cassandras, C.G. (2018). Event-driven cooperative receding horizon control for multi-agent systems in uncertain environments. *IEEE Transactions on Control of Network Systems*, 5(1), 409–422.

Li, W. and Cassandras, C.G. (2006). A cooperative receding horizon controller for multivehicle uncertain environments. *IEEE Transactions on Automatic Control*, 51(2), 242–257.

Salazar, M., Rossi, F., Schiffer, M., Onder, C.H., and Pavone, M. (2018). On the interaction between autonomous mobility-on-demand and public transportation systems. *arXiv preprint arXiv:1804.11278*.

Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S.H., and Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37), 13290–13294.

Swaszek, R. and Cassandras, C. (2019). Load balancing in mobility-on-demand systems: Reallocation via parametric control using concurrent estimation. *Proc. of 22nd IEEE Intl. Conference on Intelligent Transportation Systems*, 2148–2153.

Tsao, M., Iglesias, R., and Pavone, M. (2018). Stochastic model predictive control for autonomous mobility on demand. *arXiv preprint arXiv:1804.11074*.