

Fair Planning for Mobility-on-Demand with Temporal Logic Requests

Kaier Liang and Cristian-Ioan Vasile

Abstract—Mobility-on-demand systems are transforming the way we think about the transportation of people and goods. Most research effort has been placed on scalability issues for systems with a large number of agents and simple pick-up/drop-off demands. In this paper, we consider fair multi-vehicle route planning with streams of complex, temporal logic transportation demands. We consider an approximately envy-free fair allocation of demands to limited-capacity vehicles based on agents’ accumulated utility over a finite time horizon, representing for example monetary reward or utilization level. We propose a scalable approach based on the construction of assignment graphs that relate agents to routes and demands, and pose the problem as an Integer Linear Program (ILP). Routes for assignments are computed using automata-based methods for each vehicle and demands sets of size at most the capacity of the vehicle while taking into account their pick-up wait time and delay tolerances. In addition, we integrate utility-based weights in the assignment graph and ILP to ensure approximate fair allocation. We demonstrate the computational and operational performance of our methods in ride-sharing case studies over a large environment in mid-Manhattan and Linear Temporal Logic demands with stochastic arrival times. We show that our method significantly decreases the utility deviation between agents and the vacancy rate.

I. INTRODUCTION

With the development of urbanization, the demand for transporting people and goods is expanding. Yet simply increasing the number of private vehicles is inefficient for road traffic and not environmental-friendly. On the other hand, the mobility-on-demand system can be economical and sustainable. This system allows passengers to specify their demands and employ a large scale of ride-sharing on the road map, thus reducing the traveling cost, alleviating the traffic congestion and emission [1], [2], [3]. However, less attention has been paid to how fair transportation requests are distributed to drivers in mobility-on-demand systems.

A wealth of research has investigated the system and has focused on the real-time route planning and scalability issues with a large number of agents. Rebalance policies for the congestion and high demand were studied in [4], [5], [6]. In mesoscopic optimization, using the estimations of traffic congestion, joint operations for autonomous vehicles fleet was studied in [7], [8]. From a microscopic perspective, the requests assignment with a defined cost can be formulated as an optimization problem [9]. One approach is to construct a shareability graph between vehicles and requests for the ride-sharing [10], [11]. Based on this approach, Alonso-Mora et al. create a Requests-Trip-Vehicle (RTV) assignment graph [12]. The large ride-sharing problem is encoded using integer linear programming (ILP) and solved almost in real-time.

However, among the studies above, the requests mainly consisted of simple atomic tasks, such as driving from point A (pick-up location) to point B (drop-off location). This leaves an unexploited scenario where the requests are complex. For example, a customer may want to purchase a gift from store A or store B, while another customer wants dinner at a restaurant near store A. Suppose both of them have close pick-up positions and send out the requests at a similar time; one vehicle may be able to accommodate both of them by driving to the restaurant and store A, should the waiting and delay times be acceptable for them. This kind of request can be represented using linear temporal logic (LTL), which is employed in [13] for a single-vehicle routing to tackle complex requests assignment. The map and vehicles are modeled as weighted transition systems (WTS), and the demands are formulated using co-safe LTL (scLTL). Then we graph search algorithms check the ride-sharing feasibility and make the assignment based on the defined cost function.

Furthermore, research into ride-sharing has often focused on the customer side. The objective of requests assignment and route planning is to minimize travel costs and fairness is usually considered from the customers’ perspective [14], [15]. However, drivers’ preferences may not agree with the assignment they received. Moreover, the demand for drivers may exceed the number available, e.g., during peak hours, which can give drivers an edge in the request-driver relationship, as drivers can have more choice. Also, there might be some vacant vehicles in the assignment when there is less demand for vehicles during off-peak. Therefore, fairness should also be considered from the drivers’ perspective when allocating requests to tackle the utility disparity among drivers. There are different criteria to judge the fairness, for example, maximizing the minimum utility for vehicles [16], [17].

The contributions of this work are the following 1) we propose a multi-vehicle routing problem with fairness constraints on the assignment of temporal logic demands; the arrival time of demands is a priori unknown; we consider fairness in sequential decision making. 2) we propose a combined automata and ILP-based approach that decomposes the problem into a set of small routing problems with scLTL specifications. 3) we propose a weighting scheme for the assignment graph that corrects the history of utility collected by vehicles. 4) we show the performance of our approach in case studies on part of mid-Manhattan (Fig. 1); and we show that our approach significantly reduces the deviation of collected utility between vehicles, and the vacancy rate with respect to baseline without fairness considerations.

II. PRELIMINARIES

In this section, we introduce the notation used in the paper and review concepts in formal language and automata theory.

Kaier Liang and Cristian-Ioan Vasile are with the Mechanical Engineering and Mechanics Department at Lehigh University, PA, USA: {ka1221, cvr519}@lehigh.edu



Fig. 1: The road network corresponding to part of mid-Manhattan is shown. Travel duration estimates are inferred from real taxi travel data in hourly increments [12].

We denote the set of real and integer numbers as \mathbb{R} and \mathbb{Z} . The real and integer numbers greater than a are denoted by $\mathbb{R}_{>a}$ and $\mathbb{Z}_{>a}$. Similarly, we have $\mathbb{R}_{\geq a}$ and $\mathbb{Z}_{\geq a}$ for real and integer numbers greater or equal than a . For a finite set S , we denote its cardinality and the power set as $|S|$ and 2^S .

Definition 1 (Finite Automaton). A deterministic finite state automaton (DFA) is a tuple $\mathcal{A} = (Q_{\mathcal{A}}, q_{init}^{\mathcal{A}}, 2^{\Pi}, \delta_{\mathcal{A}}, F_{\mathcal{A}})$, where $Q_{\mathcal{A}}$ is a finite set of states; $q_{init}^{\mathcal{A}} \in Q$ is the initial state; 2^{Π} is the input alphabet; $\delta_{\mathcal{A}} : Q_{\mathcal{A}} \times 2^{\Pi} \rightarrow Q_{\mathcal{A}}$ is a transition function; $F_{\mathcal{A}} \subseteq Q_{\mathcal{A}}$ is the set of accepting states.

An input word $\sigma = \sigma_0\sigma_1 \dots \sigma_n$ over alphabet 2^{Π} generates the trajectory of the DFA $\mathbf{q} = q_0q_1 \dots q_n$ with $q_{init} = q_0$ and $q_{k+1} = \delta_{\mathcal{A}}(q_k, \sigma_k)$, for all $k \in \{0, \dots, n-1\}$. The trajectory \mathbf{q} is called *accepting* if $q_n \in F_{\mathcal{A}}$.

Definition 2 (scLTL). A co-safe Linear Temporal Logic (scLTL) formula over a set of atomic propositions Π is defined recursively as:

$$\phi ::= \pi \mid \neg\pi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \phi_1 \mathcal{U}\phi_2 \mid \diamond\phi,$$

where ϕ_1, ϕ_2 are scLTL formula, $\pi \in \Pi$ is an atomic proposition, \neg (negation), \wedge (disjunction), and \vee (conjunction) are Boolean operators, and \mathcal{U} (until), \bigcirc (next), and \diamond (eventually) are temporal operators.

The semantics of scLTL formulae are defined over infinite words with symbols from 2^{Π} . Intuitively, $\bigcirc\phi$ holds if ϕ is true at the next position in the word; $\phi_1 \mathcal{U}\phi_2$ expresses that ϕ_1 is true until ϕ_2 becomes true; and $\diamond\phi$ expresses that ϕ becomes true at some future position in the word. The formal definition of the semantics can be found in [18]. Given a word σ over the alphabet 2^{Π} that satisfies the scLTL formula ϕ , we denote the satisfaction as $\sigma \models \phi$. A finite word σ satisfies scLTL formula ϕ if for all infinite σ' the concatenated (infinite) word $\sigma\sigma' \models \phi$. The finite word σ is *minimal* if none of its prefixes satisfies ϕ .

scLTL formulae can be translated to DFAs using off-the-shelf tools such as scheck [19] and spot [20].

Definition 3 (Weighted Transition System). A weighted transition system (WTS) is a tuple $\mathcal{T} = (S, s_{init}, D, W, \Pi, L)$, where S is a finite set of states, $s_{init} \in S$ is the initial state,

$D \subseteq S \times S$ is a transition function, $W : D \rightarrow \mathbb{R}_+$ is a weight function, Π is a set of atomic propositions and $L : D \rightarrow 2^{\Pi}$ is a labeling function.

The transition from the current state s at time t to the next state s' is reached at time $t' = t + W((s, s'))$ if $(s, s') \in D$. A trajectory of \mathcal{T} is a finite sequence $\mathbf{s} = s_0s_1 \dots s_n$, such that $s_0 = s_{init}$, and $(s_k, s_{k+1}) \in D$ for all $k \in \{0, \dots, n-1\}$. The length of the trajectory \mathbf{s} is n , and its total duration is $W(\mathbf{s}) = \sum_{i=0}^{n-1} W((s_i, s_{i+1}))$. The output trajectory induced by \mathbf{s} is $\mathbf{o} = L(s_0)L(s_1) \dots L(s_n)$. A finite trajectory \mathbf{s} satisfies a scLTL formula ϕ , denoted $\mathbf{s} \models \phi$, if the induced output trajectory \mathbf{o} satisfies ϕ .

III. PROBLEM FORMULATION

In this section, we formulate the fair mobility-on-demand problem with requests expressed as scLTL specifications and vehicle sharing. Our goal is to compute assignments of sequentially incoming scLTL requests to a fleet of vehicles such that the total traveling cost is minimized, and fairness among the drivers over the planning horizon is ensured.

A. Vehicle, Environment, and Request Models

Consider a fleet of vehicles $\mathcal{V} = \{v_1, v_2, \dots, v_p\}$ deployed in a road network with intersections S and roads $D \subseteq S \times S$, where $(s, s') \in D$ represents a road from intersection s to s' . The initial position of vehicle $v \in \mathcal{V}$ is $s_{init,v} \in S$. All vehicles evolve in discrete time $t \in \mathbb{Z}_{\geq 0}$ synchronized via a global clock. The traversal duration of road (s, s') is $W((s, s')) \in \mathbb{Z}_{>0}$.

Vehicles are tasked with satisfying a finite set of request $\mathcal{R} = \{r_1, r_2, \dots, r_m\}$ that arrive sequentially over the horizon time $H \in \mathbb{Z}_{>0}$. A request $r \in \mathcal{R}$ is defined as a tuple $r = (\pi_{pick,r}, \phi_r, t_{req,r}, \rho_r, \Omega_{max,r}, \Delta_{max,r})$, where

- $\pi_{pick,r}$ is a proposition marking the pick-up location;
- ϕ_r is the scLTL formula specifying the request;
- $t_{req,r} \in \{0, \dots, H\}$ is the request's arrival time;
- $\rho_r \in \mathbb{Z}_{>0}$ is the number of required seats;
- $\Omega_{max,r} \in \mathbb{Z}_{>0}$ is the maximum waiting time, i.e., the latest accepted pick-up time is $t_{req,r} + \Omega_{max,r}$;
- $\Delta_{max,r} \in \mathbb{Z}_{>0}$ is the maximum allowed delay.

Vehicles have limited transportation capacities. We denote by $Cap_v \in \mathbb{Z}_{>0}$ and $c_v(t) \in \{0, \dots, Cap_v\}$ the maximum capacity and the available capacity at time t for vehicle $v \in \mathcal{V}$. A vehicle v is said to be *available* at time t if $c_v(t) > 0$, otherwise it is *occupied*, i.e., $c_v(t) = 0$. The set of available vehicles at time t is denoted by \mathcal{V}_t^a .

A group of vehicles $V \subseteq \mathcal{V}$ completes a request $r \in \mathcal{R}$ if they pick up r at the intersection marked with $\pi_{pick,r}$ such that their overall available capacity is greater than ρ_r . Formally, we have $\mathbf{s}_v \models \phi_r$, vehicle v is available at time $t_{pick,r,v}$ for all vehicles $v \in V$, and $\sum_{v \in V} c_v(t_{pick,r,v}) \geq \rho_r$, where $\phi_r = \diamond(\pi_{pick,r} \wedge \phi_r)$, \mathbf{s}_v is the finite trajectory of v and $t_{pick,r,v}$ is the pick-up time for r by v . Note that we do not require all vehicles V to pick up their share of request r at the same time.

The delay Δ_r is the difference between the actual and optimal satisfaction duration. Formally, $\Delta_r = \max_{v \in V} t_{drop,r,v} - t_{req,r} - t_r^*$, where $t_{drop,r,v}$ is the drop off time of request r by

vehicle v . $(s_v(0:t_{drop,r,v}))$ is a minimal satisfying word for $\tilde{\phi}_r$ and t_r^* is the optimal satisfaction time, i.e., the amount of travel time if a vehicle picks up the request at $t = t_{req,r}$ and not share with other requests. We require that $\Delta_r \leq \Delta_{max,r}$.

At current time $t \in \mathbb{Z}_{\geq 0}$, a request is *active* if $t_{req} \leq t$ and it has not been picked-up yet; a request is *in progress* if it has been picked-up and not completed. The sets of active and in progress requests at time t are \mathcal{R}_t^a and \mathcal{R}_t^p , respectively.

Example 1. A small road map in WTS form with the set of atomic proposition $\Pi = \{A, \dots, F\}$ is depicted in Fig. 2. The pick-up locations $\pi_{pick,1}$ and $\pi_{pick,2}$ shown in red dots at C and B represent requests with scLTL formulas specifying as $\phi_1 = \diamond(D \wedge \diamond E)$ and $\phi_2 = \diamond(D \wedge \diamond F)$, respectively. The blue dot represents the initial position of an empty vehicle v_1 . The least travel times are $t_1^* = 9$ and $t_2^* = 5$ and the assignment planning result is shown in Table. I.

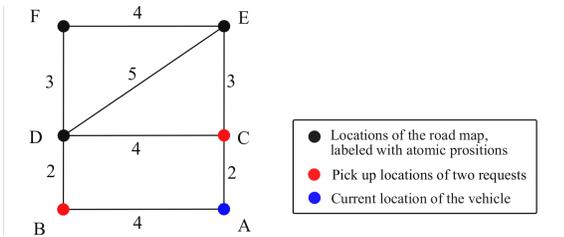


Fig. 2: Example of road map model as a WTS, the drivable paths are labeled with weights as traveling cost between each connected nodes. For an empty vehicle at location A , assuming both requests are active and the maximum delay and waiting time are satisfied, the route of the vehicle v_1 with minimal travel cost is $A \rightarrow C \rightarrow D \rightarrow B \rightarrow D \rightarrow F \rightarrow E$.

An assignment $Asg_t : \mathcal{R}_t^a \rightarrow 2^{\mathcal{V}^a}$ at time $t = t_{req,r}$ allocates active requests to vehicles when request r arrives. If the assignment $Asg_t(r) = \emptyset$, then r is *unassigned* at time t . In case this holds for all $t \in \{t_{req,t}, \dots, t_{req,r} + \Omega_{max,r}\}$, r is unassigned. An assignment for r may involve multiple vehicles, i.e., $|Asg_t(r)| > 1$. Requests that are *in progress* cannot be reassigned and vehicles need to be *available* before picking up new requests. Between request arrivals times, i.e., $t \neq t_{req,r}$, assignments do not change.

The total cost for all requests is defined as

$$J(\{Asg_t\}_{t=0}^H, \{s_v\}_{v \in \mathcal{V}}) = \sum_{i=1}^m \Delta_{r_i} + \lambda_{ko} |\Upsilon|, \quad (1)$$

where $\Upsilon = \{r \in \mathcal{R} \mid Asg_t(r) = \emptyset, \forall t \geq t_{req,r}\}$ is the set of unassigned requests, and $\lambda_{ko} > 0$ is a penalty for not fulfilling a request. The cost depends on the requests assigned to vehicles, and the routes computed to complete them.

B. Envy-Free Fairness

The cost J captures customer satisfaction (performance of the mobility-on-demand system). However, it is equally important to consider fairness in allocating requests from the drivers' perspective. We formalize the notion of utility for vehicles, and impose envy-free division [21] of requests over finite time horizons.

Let $\Gamma_v \subseteq \mathcal{R}$ be set of requests completed by vehicle v . The utility of vehicle v with maximum capacity Cap_v is

$$U_v(\Gamma_v) = \sum_{t=0}^H (Cap_v - c_v(t)), \quad (2)$$

and captures the utilization of v over the time horizon H .

The request assignment over the time horizon H is called *envy-free* if $U_v(\Gamma_v) \geq U_{v'}(\Gamma_{v'})$, for all $v, v' \in \mathcal{V}$. Due to the sequential arrival of requests, we can not impose the envy-free condition over the total utility in the time horizon H . Instead, we investigate the slightly weaker condition that the vehicles' utilities are envy-free when re-computing assignments at requests' arrival times.

Problem 1 (Fair Request Assignment). *Given the set of vehicles \mathcal{V} deployed in environment (S, D, W) , and the set of requests $\mathcal{R} = \{r_1, \dots, r_m\}$ arriving sequentially over time horizon H , compute assignments Asg_t at each time $t \in \{0, \dots, H\}$ and routes s_v for all vehicles $v \in \mathcal{V}$ such that the vehicles' utilities satisfy the envy-free fairness conditions and minimizes the cost J .*

Summary of the approach. When a new request arrives, or a vehicle becomes available, We construct an assignment graph to match requests and vehicles. The RTV graph has three layers (1) requests, (2) trips, and (3) vehicles. Edges that connect vehicles to trips serving a subset of active requests are computed via an automata-based routing procedure. We construct product automata between the motion model (road network) of a vehicle, and the DFAs corresponding to the requests. The route is then computed via a shorted path method (e.g., Dijkstra algorithm) applied on the product automaton graph and projection onto the motion model. If maximum waiting and delay times constraints are met, we add the edge to the assignment graph. Lastly, we can formulate an ILP problem to minimize the sum of travel costs such that the envy-free constraints hold for the allocated utilities of each vehicle. The solution of the ILP provides the assignment scheme.

IV. SOLUTION

The mobility-on-demand ride-sharing problem can be translated to an ILP problem through the construction of a shareability graph and an assignment (RTV) graph [10], [12]. Then we can apply graph search algorithms and provide efficient solutions. This paper uses automata theory to construct the assignment graph, and applies fair planning through envy-free constraints and a proposed graph weight correction method. In the following, we drop the time subscript t to improve readability, and whenever it is clear from context.

A. Request-Trip-Vehicle (RTV) Graph

The RTV graph batch assignment was introduced in [12]. First, we construct the undirected Request-Vehicle (RV) graph $\mathcal{G}^{RV} = (\mathcal{R}^a \cup \mathcal{V}^a, E^{RV})$ that captures requests that may be performed by a vehicle in a single trip without violating the waiting time, and maximum delay constraints. The RV graph's nodes are the requests and vehicles. Edges $e(r, r')$ between two requests capture their shareability, i.e., can be

TABLE I: Example for requests in Figure. 2

	Pick-up Location	scLTL spec	Arrival Time	Pick-up time	Drop-off Time	Delay
r_1	$s_{pick,1} = C$	$\diamond(D \wedge \diamond E)$	$t_{req,1} = 0$	$t_{pick,1} = 2$	17	8
r_2	$s_{pick,2} = B$	$\diamond(D \wedge \diamond F)$	$t_{req,2} = 0$	$t_{pick,2} = 8$	13	8

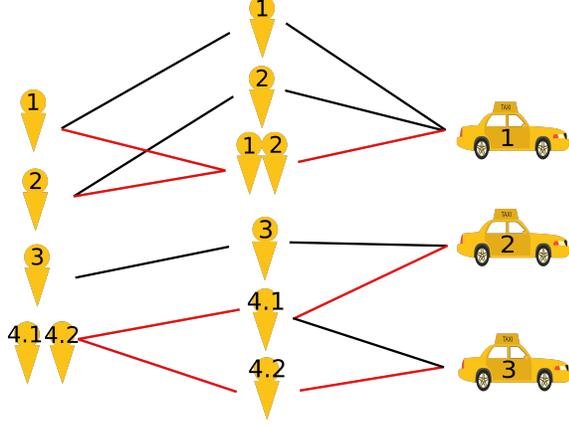


Fig. 3: Example of RTV graph: The graph includes 4 requests and 3 vehicles. The fourth requests requires two vehicles. The red edges indicate a possible assignment scheme.

served at the same time by a vehicle. Edges $e(v, r)$ indicate whether vehicle v can serve request r under the required timing constraints.

Next, we construct the undirected RTV graph \mathcal{G}^{RTV} with nodes $\mathcal{R}^a \cup \text{Tr} \cup \mathcal{V}$ and edges E^{RTV} , where Tr is the set of trips, see Fig. 3. A trip $T_v \subseteq \mathcal{R}^a$ is a subset of requests serviced by a vehicle v . Multiple vehicles v_1, v_2, \dots, v_{n_r} may be needed to service a single request r , in which case the r is part of all their trips T_{v_i} , for all $i \in \{1, \dots, n_r\}$. Trips are formed from the RV graph by selecting its cliques [12] that satisfy timing and capacity constraints for vehicles. Thus, the RTV graph contains only potentially feasible trips of active requests for available vehicles. Edges $e(r, T) \in E^{RTV}$ denote request r is part of trip T , while edges $e(T, v)$ denote that v can serve requests in trip T , see Fig. 3.

Next, we define procedures to decide if edges $e(v, r)$ and $e(v, T)$ belong to the RV graph \mathcal{G}^{RV} and the RTV graph \mathcal{G}^{RTV} , respectively, for all $v \in \mathcal{V}^a$, $r \in \mathcal{R}$, and $T \in \text{Tr}$.

B. Automata-based Route Planning

We construct product automata to obtain the RTV graph for scLTL requests and vehicles represented as a transition system (TS). Formally, we have the TS $\mathcal{T}_v = (S, s_{init}, D, W, \Pi, L)$ that captures vehicle v 's motion in the environment. The set of propositions Π includes the active requests' pick-up propositions $\pi_{pick,r}$.

Definition 4 (Weighted product automaton at time t). *The weighted product automaton $\mathcal{P} = \mathcal{T} \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_m$ of vehicle v at time t is a tuple $(Q_{\mathcal{P}}, Q_{init,\mathcal{P}}, \delta_{\mathcal{P}}, F_{\mathcal{P}}, W_{\mathcal{P}})$, where*

- $Q_{\mathcal{P}} = \{s, q_1, \dots, q_m\}$;

Algorithm 1: Fair Request Assignment Algorithm

Input: \mathcal{R}^a – the active requests, \mathcal{V}^a – the available vehicles

Output: $\text{Asg} : \mathcal{V}^a \rightarrow \text{Tr}$ – vehicles to trips assignment

```

// Construct RV Graph
1  $\mathcal{G}^{RV} = (\mathcal{R}^a \cup \mathcal{V}^a, E^{RV} = \emptyset)$ 
2 forall  $r, r' \in \mathcal{R}^a, r \neq r'$  do
3   if check_share( $r, r'$ ) then  $E^{RV} \leftarrow E^{RV} \cup e(r, r')$ 
4 forall  $r \in \mathcal{R}^a, v \in \mathcal{V}^a$  do
5   if check_trip( $v, \{r\}$ ) then  $E^{RV} \leftarrow E^{RV} \cup e(r, v)$ 
// Construct RTV Graph
6  $\text{Tr} \leftarrow$  all the cliques of requests in  $\mathcal{G}^{RV}$  that satisfy timing
  and capacity constraints
7  $\mathcal{G}^{RTV} = (\mathcal{R}^a \cup \text{Tr} \cup \mathcal{V}^a, E^{RTV} = \emptyset)$ 
8  $E^{RTV} \leftarrow \{(r, T) \mid r \in T\}$ 
9 forall  $T \in \text{Tr}, v \in \mathcal{V}^a$  do
10  if check_trip( $v, T$ ) then  $E^{RTV} \leftarrow E^{RTV} \cup e(T, v)$ 
11  $\text{Asg} = \text{solve\_ILP}(\mathcal{G}^{RTV})$ 
12 return  $\text{Asg}$ 

```

- $Q_{init} = \{s_j, \pi_{pick,1}, \dots, \pi_{pick,m}\}$, where s_j is the current node of the v_k in the map, i.e., $s_0 = s_{init}$;
- $q_{i,j} = \begin{cases} \delta_i(\pi_{pick,i}, L(s_j)) & \text{if } t_{pick,r_i} = t \\ \delta_i(q_{i,j-1}, L(s_j)) & \text{if } t_{pick,r_i} < t \\ q_{init,i} & \text{else,} \end{cases}$, where t_{pick,r_i} is the pick-up time for r_i ;
- $\delta_{\mathcal{P}} \subseteq Q_{\mathcal{P}} \times Q_{\mathcal{P}}$ is a transition function: $((s, q_1, \dots, q_m), (s', q'_1, \dots, q'_m)) \in \delta_{\mathcal{P}}$ if and only if $(s, s') \in R$ and $(q_i, L(s'), q'_i) \in \delta_i$;
- $F_{\mathcal{P}} = \{(s, q_{1,k}, \dots, q_{m,k}) \mid q_{i,k} \in F_i, \forall i \in \{1, \dots, m\}\}$;
- $W_{\mathcal{P}} : \delta_{\mathcal{P}} \rightarrow \mathbb{R}_+$ is the weight function given by $W_{\mathcal{P}}(((s, q_1, \dots, q_m), (s', q'_1, \dots, q'_m))) = W(s, s')$.

1) *Weighted product automaton for pairwise request-request in RV graph (check_share):* This step checks if two requests r and r' can potentially be shared by the same vehicle, i.e., check_share procedure used in Alg. 1. Two requests can be combined pairwise if a virtual vehicle starting at one of their pick-up positions can complete both requests, i.e., satisfy the maximum delay and maximum wait time of both requests. To achieve this, we construct a weighted automaton $\mathcal{P}_{RR} = \mathcal{T}_{virtual} \otimes \mathcal{A}_r \otimes \mathcal{A}_{r'}$ for r and r' . $\mathcal{T}_{virtual}$ is the transition system for the virtual vehicle with initial position $s_{init,virtual} \in \{q_{init,r}, q_{init,r'}\}$. Then, we use a graph search method such as Dijkstra's algorithm to check if an admissible path exists [22]. If it exists, edge $e(r, r')$ is added to the RV graph. Moreover, these two requests are a potential candidate for a trip $T = \{r, r'\}$ denoted as edges $e(r, T)$ and $e(r', T)$ in the RTV graph. For example,

in Fig. 2, the trip $T_k = (r_1, r_2)$ with corresponding edges $e(r_1, T_k)$ and $e(r_2, T_k)$ is added to the RTV graph.

2) *Weighted product automaton for pairwise request-vehicle in RV graph* (check_trip with $|T| = 1$): The construction of the weighted product automaton for request-vehicle combination is similar to the product automaton for request-request. For every available vehicle v and request r , we construct a weighted product automaton $\mathcal{P}_{RV} = \mathcal{T}_v \otimes \mathcal{A}_r$. The difference in this product automaton is that real-time vehicle information, i.e., the vehicle's position, is used. Likewise, v and r are connected in the RV graph via edge $e(r, v)$ if an admissible path is found in the product automaton. For example, edges $e(r_1, v_1)$ and $e(r_2, v_1)$ are in the RV graph for the case shown in Fig. 2.

3) *Weighted product automaton for RTV graph* (check_trip with $|T| > 1$): The connected requests and vehicles in the RV graph are feasible candidates for an assignment in the RTV graph with trips containing only one request, and the ride-sharing trips with more than one request can be built based on the RV graph. For requests r, r' and vehicle v_i , if the pair (r, r') is present in the RV graph, this means these two requests can share a vehicle, in the best-case scenario, when the vehicle is at their pick-up positions. And if (r, v_i) and (r', v_i) are also present in the RV graph, this means v_i can serve r or r' under no sharing condition. If both these conditions are satisfied, we can further validate the ride-sharing possibility of v_i to serve both r and r' by constructing a weighted product automaton $\mathcal{P}_{RTV} = \mathcal{T}_i \otimes \mathcal{A}_r \otimes \mathcal{A}_{r'}$. And if an admissible path is found without violating the request constraints, v_i and $T_j = \{r, r'\}$ are grouped as a potential valid trip in the assignment and an edge $e(T_j, v_i)$ is created in the RTV graph to denote a potential assignment (T_j, v_i) . For example $e(T_k, v_1), T_k = \{r_1, r_2\}$ would be created for Fig. 2. The RTV graph can be generated recursively for $Cap_v \geq 2$. In this paper, we consider the case where $Cap_v = 2$.

In the check_trip step, for a vehicle v_i and allocated trip $T_j = \{r_1, \dots, r_n\}$, the travel cost $\sigma_{v_i}(T_j)$ and travel utility $U_{v_i}(T_j)$ associated with each created edge are simultaneously generated as

$$\begin{aligned} \sigma_{v_i}(T_j) &= \sum_{i=0}^n \Delta_{r_i}, \\ U_{v_i}(T_j) &= \sum_{t=0}^h (Cap_{v_i} - c_{v_i}(t)), \end{aligned} \quad (3)$$

where $n = |T_j|$ and h is the trip serving duration.

4) *Weight Correction Based on History Utility*: One problem of non-fair assignment is that it doesn't consider the history utility, which may create a significant vacancy rate or disparity of total utility. For example, during an off-peak hour, there might be a lesser number of requests than the number of vehicles available. Thus, some vehicles may not ever be allocated to any trips or only assigned with a low utility trip. Therefore, we make a cost correction based on a history utility to balance the accumulated utility over time.

For a RTV graph, at any time step, the travel cost associated with edge $e(T_j, v_i)$ is adjusted in the following way

$$\sigma_{v_i}^{new}(T_j) = \sigma_{v_i}^{old}(T_j) + \alpha \cdot (U_{v_i} - U_{avg}), \quad (4)$$

where $\alpha \in \mathbb{R}_{>0}$ is a constant parameter and $U_{avg} = \sum_{i=1}^p U_{v_i}/p$, $p = |\mathcal{V}|$ is the average history utility for all vehicles. After the weight correction, the traveling cost decreases for vehicles with low history utility and increases for vehicles with high history utility, thus favoring trips for vehicles with low history utility.

C. ILP Formulation

This section describes the solve_ILP function in Algorithm 1. We formulate the vehicle-sharing and fairness problem using ILP, which needs to be updated when a new request arrives, or a vehicle becomes available.

A binary variable $\epsilon_{i,j} \in \{0, 1\}$ is introduced for each edge $e(T_j, v_i)$ in the RTV graph, $\epsilon_{i,j} = 1$ indicates that vehicle v_i is assigned to trip T_j . In addition, a binary variable $\chi_k \in \{0, 1\}$ is introduced for each request r_k . If χ_k takes the value one, it means that request r_k is not served by any vehicles.

For multiple vehicles serving a single request, for compatibility with the bipartite graph representation, we divide the original request ρ_i into j sub-requests with each $\rho_{i,j} = 1$ and $\sum \rho_{i,j} = \rho_i$. For example, ϕ_i is written as $\phi_{i,1} = \diamond(\text{store 1})$ and $\phi_{i,2} = \diamond(\text{store 2})$ with the same t_{req} . Then, we constrain the assignment to contain either zero or all sub-requests.

The objective of the ILP is to minimize the assignment cost. The ILP formulation is defined as:

$$\min \sum_{(i,j): e(T_i, v_j)} \sigma_{v_i}(T_j) \epsilon_{i,j} + \sum_{k=1}^m \lambda_{ko} \chi_k, \quad (5a)$$

$$\text{s.t.} \sum_{i: e(R_k, T_i)} \sum_{j: e(T_i, v_j)} \epsilon_{i,j} + \chi_k = 1, \quad \forall r_k \in \mathcal{R}, \quad (5b)$$

$$\sum_{i: e(T_i, v_j)} \epsilon_{i,j} \leq 1, \quad \forall v_j \in \mathcal{V}, \quad (5c)$$

$$U_{v_i}(T_m) - \lambda \cdot U_{v_j}(T_n) \geq M(\sigma_{v_i}(T_m) + \sigma_{v_j}(T_n) - 2) \quad (5d)$$

The cost function defined in (5a) minimizes the sum of travel cost plus a penalty λ_{ko} for every unassigned request. (5b) and (5c) indicate each request is assigned to one vehicle at most, and each vehicle is assigned to one trip at most. The envy-free fairness constraint is captured by constraint in (5d) using the *big M* method, where M is a constant value that is larger than the maximum value of all trip utilities.

Note that we do not need to compare the utilities of every pair of vehicles at a time step. For example, T_m may be infeasible for v_j in (5d), thus lending the comparison trivial. Moreover, some vehicles may not be available at t .

Note that in a strict envy-free allocation, the resulting matching may be undesirable in some scenarios. For example, suppose in a road network containing two vehicles v_1 and v_2 and two requests r_1 and r_2 , both v_1, v_2 can serve r_1 and only v_1 can serve r_2 . Ideally, the optimal solution matching pair is (v_1, r_2) and (v_2, r_1) . However, if r_1 has larger utility than r_2 for vehicle v_1 , then the envy-free matching would only allocate (v_1, r_2) leaving r_1 vacant and v_2 unoccupied. Otherwise, if either vehicle serves r_1 , then they would envy the other.

To tackle the problem, we adapt the relaxation idea *envy-free up to one item*. Two agents would not envy each other if one item is removed from the environment [23].

However, the assignment is indivisible and one vehicle can only be allocated to one assignment at a time, and as such removing one item is not feasible. Thus, we introduce a variable $\lambda \in [0, 1]$ to regulate the approximation of envy-free to (5d). When $\lambda = 0$ the envy-free constraints is disabled, and $\lambda = 1$ enforces strict envy-free. In the previous example, if $U_{v_1}(r_2) \geq \lambda \cdot U_{v_2}(r_1)$ holds for $\lambda \leq \frac{U_{v_1}(r_2)}{U_{v_2}(r_1)}$, the envy-free allocation is (v_1, r_2) and (v_2, r_1) .

This approximately envy-free approach can still suffer from the previous issue in extreme scenarios, for example, when there is significant utility disparity between the two available requests, making the relaxation λ be close to 0 to obtain the optimal overall utility. And because of a large number of requests and long working period, the total utility deviation among vehicles can be similar regardless of the envy-free enforced.

A greedy solution to maximize the serving rate while minimizing the travel cost is first computed as an initial guess for the ILP. Once solving the above ILP problem, the assignment scheme constructs the optimal path for vehicles as the accepted and shortest run in the corresponding product automaton projected onto the transition system.

V. SIMULATION RESULTS

In this section, we present simulation results to demonstrate the performance in terms of scalability and fairness in a realistic road map.

A. Simulation Specifications

We simulated the result in the mid-Manhattan map, which models the road intersections as nodes. The map contains 184 nodes, and the edges are weighted by real travel duration obtained from real taxi driving data. For details about the dataset, see [12]. Gurobi was used to solve the ILP [24]. The simulation duration is set to 20 minutes with varying the number of vehicles and requests. The scLTL formulas were generated from the following scLTL pattern stochastically.

scLTL pattern:

$$\tilde{\phi}_1(s_{pick}, s_1, s_2) = \diamond(s_{pick} \wedge \diamond(s_1 \wedge \diamond(s_2))),$$

$$\tilde{\phi}_2(s_{pick}, s_1, s_2) = \diamond(s_{pick} \wedge \diamond((s_1 \vee s_2) \wedge s_3)),$$

$$\tilde{\phi}_3(s_{pick}, s_1, s_2, s_3) = \diamond(s_{pick} \wedge \diamond(s_1 \wedge (s_2 \vee s_3))),$$

$$\tilde{\phi}_4(s_{pick}, s_1, s_2, \dots, s_n) = \diamond(s_{pick} \wedge \diamond(s_1 \wedge (\neg s_2 \wedge \dots \wedge \neg s_n))),$$

where s_i are locations in the road map. The multi-vehicles serving requests are combinations of the above scLTL pattern echoing the sub-requests division technique. Furthermore, we also generate random pick-up positions and arrival times for each request in a uniform Poisson process. The maximum waiting time and delay time are set to $\Omega_{max} = 2$ and $\Delta_{max} = 4$ minutes for every request, respectively.

The vehicle's transportation capacity is set to at most two requests at a time for ride-sharing, i.e., $Cap_v = 2$, and we generate the initial positions for the vehicles at time $t = 0$ stochastically. The envy-free variable λ is set to 0.5.

B. Simulation Results and Discussions

We simulate the results by varying the vehicle-to-request ratio to demonstrate fairness and run time performance.

In the simulation shown in Fig. 4 we consider 50 vehicles and 100 requests in the network. The figure shows the steady serving pace along with the gradually arriving requests.

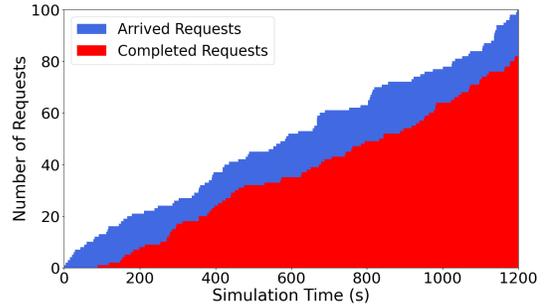


Fig. 4: Number of arrived and completed requests over time

Fig. 5 shows the comparison between fair planning versus non-fair planning, i.e., plan without envy-free constraints and weight correction. Each data point in the figure is the average data of ten runs. We study the vacancy rate and utility deviation as fair criteria.

The vacancy rate is defined as the percentage of unoccupied vehicles in the entire simulation time. Intuitively, the higher ratio of the vehicle to request, the higher the vacancy rate, as shown in Fig. 5a. Under fair planning conditions, the vacancy rate is significantly reduced.

In addition, Fig. 5b shows the history of utility deviation. The figure shows that fair planning also significantly decreases the utility deviation among vehicles. The result of Fig. 5 is expected as the fair planning makes the ILP solution favor vehicles with low utility and, thus, reduces the vacancy rate as well. In Fig. 5b, when there are small number of vehicles in the road, the utility deviation is similar. This is also expected as almost every vehicle would receive an assignment once it becomes available, making fair planning similar to the non-fair baseline.

Fig. 6 shows the average computational run time performance for the simulation. We fix the number of vehicles and requests in Fig. 6a and Fig. 6b to demonstrate the scalability of our approach. The construction of the road map and RTV graph contribute most of the simulation time. The results show run time is similar for a fixed number of requests, and increases with the number of requests.

VI. CONCLUSIONS AND FUTURE WORK

A fair planning mobility-on-demand with temporal logic requests study is presented in this paper. The scLTL formulated requests allow passengers to define complex requests. We employ envy-free allocation and a utility-based weight correction to achieve a fair division of requests for vehicles. We show that fair planning significantly decreases the vacancy rate and utility deviations between vehicles compared to a baseline that does not consider fairness constraints. Moreover, we show that our method scales well with the number of vehicles and requests.

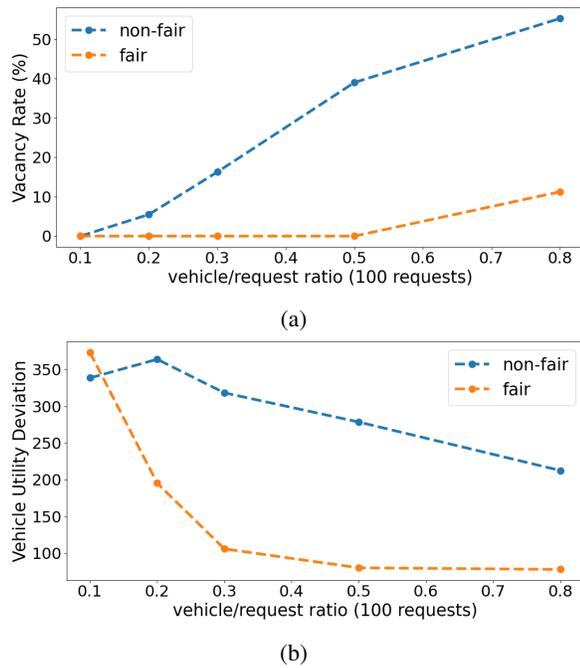


Fig. 5: Comparison between fair and non-fair planning. (a) Vehicle Vacancy Rate. (b) Vehicle Utility Deviation

REFERENCES

- [1] T. Teubner and C. M. Flath, "The economics of multi-hop ride sharing," *Business & Information Systems Engineering*, vol. 57, no. 5, pp. 311–324, 2015.
- [2] B. Caulfield, "Estimating the environmental benefits of ride-sharing: A case study of dublin," *Transportation Research Part D: Transport and Environment*, vol. 14, no. 7, pp. 527–531, 2009.
- [3] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.
- [4] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.
- [5] J. Wen, J. Zhao, and P. Jaillet, "Rebalancing shared mobility-on-demand systems: A reinforcement learning approach," in *Intl Conf on Intelligent Transportation Systems*, pp. 220–225, IEEE, 2017.
- [6] A. Wallar, M. Van Der Zee, J. Alonso-Mora, and D. Rus, "Vehicle rebalancing for mobility-on-demand systems with ride-sharing," in *Intl Conf on Intelligent Robots and Systems*, pp. 4539–4546, IEEE, 2018.
- [7] M. Salazar, M. Tsao, I. Aguiar, M. Schiffer, and M. Pavone, "A congestion-aware routing scheme for autonomous mobility-on-demand systems," in *European Control Conf*, pp. 3040–3046, IEEE, 2019.
- [8] M. Salazar, N. Lanzetti, F. Rossi, M. Schiffer, and M. Pavone, "Intermodal autonomous mobility-on-demand," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3946–3960, 2019.
- [9] D. O. Santos and E. C. Xavier, "Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6728–6737, 2015.
- [10] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13290–13294, 2014.
- [11] B. Cao, L. Alarabi, M. F. Mokbel, and A. Basalamah, "Sharek: A scalable dynamic ride sharing system," in *2015 16th IEEE International Conference on Mobile Data Management*, vol. 1, pp. 4–13, 2015.
- [12] J. Alonso-Mora, S. Samaranyake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.
- [13] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating

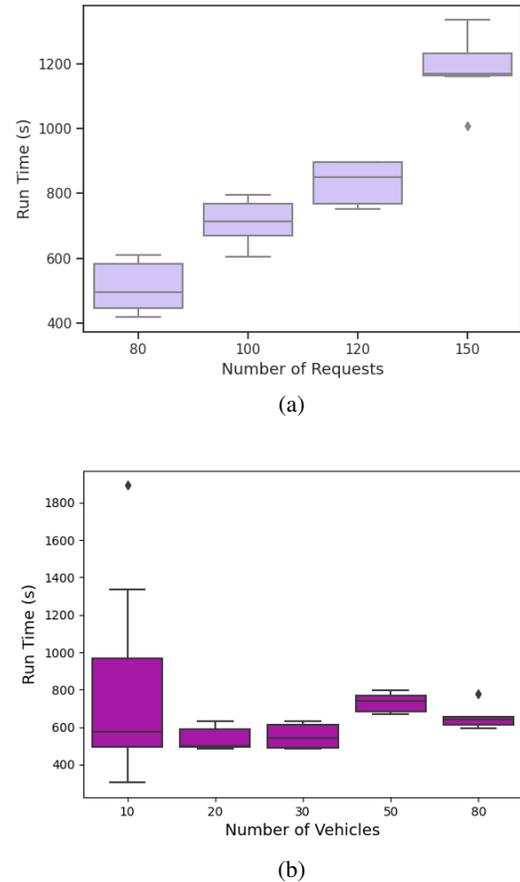


Fig. 6: Run Time Performance (a) for fixed number of 50 vehicles (b) for fixed number of 100 requests

- planning in road networks from temporal logic specifications," in *Intl Conf on Cyber-Physical Systems*, pp. 1–9, IEEE, 2016.
- [14] L. Foti, J. Lin, O. Wolfson, and N. D. Rishé, "The nash equilibrium among taxi ridesharing partners," in *ACM SIGSPATIAL Intl Conf on Advances in Geographic Information Systems*, pp. 1–4, 2017.
- [15] Y. Cao, S. Wang, and J. Li, "The optimization model of ride-sharing route for ride hailing considering both system optimization and user fairness," *Sustainability*, vol. 13, no. 2, p. 902, 2021.
- [16] O. Wolfson and J. Lin, "Fairness versus optimality in ridesharing," in *Intl Conf on Mobile Data Management*, pp. 118–123, IEEE, 2017.
- [17] N. S. Lesmana, X. Zhang, and X. Bei, "Balancing efficiency and fairness in on-demand ridesourcing," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [18] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.
- [19] T. Latvala, "Efficient Model Checking of Safety Properties," in *10th International SPIN Workshop, Model Checking Software*, pp. 74–88, Springer, 2003.
- [20] A. Duret-Lutz, "Manipulating LTL formulas using Spot 1.0," in *Intl Symposium on Automated Technology for Verification and Analysis*, vol. 8172 of LNCS, (Hanoi, Vietnam), pp. 442–445, Springer, 2013.
- [21] S. J. Brams, S. J. Brams, and A. D. Taylor, *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.
- [22] M. Sniedovich, "Dijkstra's algorithm revisited: the dynamic programming connexion," *Control and cybernetics*, vol. 35, no. 3, pp. 599–620, 2006.
- [23] E. Budish, "The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes," *Journal of Political Economy*, vol. 119, no. 6, pp. 1061–1103, 2011.
- [24] G. Optimization, "Inc., 2016. gurobi optimizer reference manual," URL <http://www.gurobi.com>.