Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Scalable and Congestion-aware Routing for Autonomous Mobility-on-Demand via Frank-Wolfe Optimization

Kiril Solovey, Mauro Salazar and Marco Pavone

Autonomous Systems Lab, Stanford University, Stanford, CA 94305, USA
{kirilsol,samauro,pavone}@stanford.edu

Abstract*—We consider the problem of vehicle routing for Autonomous Mobility-on-Demand (AMoD) systems, wherein a fleet of self-driving vehicles provides on-demand mobility in a given environment. Specifically, the task it to compute routes for the vehicles (both customer-carrying and empty travelling) so that travel demand is fulfilled and operational cost is minimized. The routing process must account for congestion effects affecting travel times, as modeled via a volume-delay function (VDF). Route planning with VDF constraints is notoriously challenging, as such constraints compound the combinatorial complexity of the routing optimization process. Thus, current solutions for AMoD routing resort to relaxations of the congestion constraints, thereby trading optimality with computational efficiency. In this paper, we present the first computationally-efficient approach for AMoD routing where VDF constraints are explicitly accounted for. We demonstrate that our approach is faster by at least one order of magnitude with respect to the state of the art, while providing higher quality solutions. From a methodological standpoint, the key technical insight is to establish a mathematical reduction of the AMoD routing problem to the classical traffic assignment problem (a related vehicle-routing problem where empty traveling vehicles are not present). Such a reduction allows us to extend powerful algorithmic tools for traffic assignment, which combine the classic Frank-Wolfe algorithm with modern techniques for pathfinding, to the AMoD routing problem. We provide strong theoretical guarantees for our approach in terms of near-optimality of the returned solution.

I. Introduction

Mobility in urban environments is becoming a major issue on the global scale [23]. The main reasons are an increasing population with higher mobility demands and a slowly adapting infrastructure [1], resulting in serious congestion problems. In addition, the usage of public transit is dropping, whilst mobility-on-demand operators such as Uber and Lyft are increasing their operation on urban roads, increasing further congestion [7, 26, 43]. For instance, the yearly cost of congestion in the US has doubled between 2007 and 2013 [39, 51], and in Manhattan cars are traveling about 15% slower compared to five years ago [17].

Space limitations and a largely fixed infrastructure make congestion an issue difficult to address in urban environments. While existing public transportation systems need to be extended to ease congestion, it is important to adopt technological innovations improving the efficiency of urban transit. The advent of cyber-physical technologies such as autonomous driving and wireless communications will enable the deployment of Autonomous Mobility-on-Demand

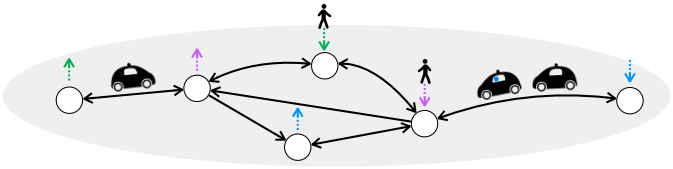

Fig. 1. The AMoD network. The white circles represent intersections and the black arrows denote road links. The dotted arrows represent pick-up and drop-off locations for single customers.

(AMoD) systems, i.e., fleets of self-driving cars providing on-demand mobility in a one-way vehicle-sharing fashion (see Fig. 1). Specifically, such a system is designed to carry passengers from their origins to their destinations, potentially in an intermodal fashion (i.e., utilizing several modes of transportation), and to assign empty vehicles to new requests. The main advantage of AMoD systems is that they can be controlled by a *central* operator simultaneously computing routes for customer-carrying vehicles and *rebalancing* routes for empty vehicles, thus enabling a *system-optimal* operation of this transportation system. This way, AMoD systems could replace current taxi and ride-hailing services and reduce the global cost of travel [46].

Conversely to conventional navigation providers computing the fastest route by passively considering congestion in an exogenous manner, AMoD systems controlled by a central operator enable one to consider the *endogenous* impact of the single vehicles' routes on road traffic and travel time, and can thus be operated in a congestion-aware fashion.

Statement of contributions: We introduce a computationally-efficient approach for congestion-aware AMoD routing to minimize the system cost—the total cost of executing the routing scheme over all the vehicles in the system. To the best of our knowledge, this is the first method that takes into consideration the full representation of the volume-delay function that estimates the travel time based on the amount of traffic. Moreover, we demonstrate that our approach is faster by at least one order of magnitude than previous work (see Section II) for congestion-aware AMoD, while being more accurate in terms of congestion estimation.

On the algorithmic side, we develop a reduction which transforms the AMoD routing problem into a Traffic Assignment Problem (TAP), where the latter does not involve rebalancing of empty vehicles. We then prove mathematically

that an optimal solution for the latter TAP instance yields a solution to our original AMoD problem with the following properties: (i) The majority (e.g., 99% in our experiments) of rebalancing demands are fulfilled and (ii) the system cost of the solution is upper bounded by the system optimum where 100% of the rebalancing demands are fulfilled. (We note that, in practice, the unfulfilled rebalancing demand, being just a small fraction – say, $< 1\%$ – can be addressed via post-processing heuristic strategies with minimal impact on cost.)

Such a reformulation of the AMoD problem allows us to leverage state-of-the-art techniques for TAP, that can efficiently compute a congestion-aware system optimum. In particular, we employ the classic Frank-Wolfe algorithm [13, 28], which is paired with modern shortest-path techniques, such as contraction hierarchies [14] (both of which are implemented in recent open-source libraries [8, 12]). This allows us to compute in a few seconds (on a commodity laptop) AMoD routing schemes for a realistic test case over Manhattan, New York, consisting of $156{,}000$ passenger travel requests.

Organization: The remainder of this paper is structured as follows. In Section II we provide a review of related work. In Section III we formally define the instances of TAP and AMoD we are concerned with in this work. There we also discuss the assumptions of our model and possible limitations. In Section IV we provide a description of the Frank-Wolfe algorithm for TAP. Our main theoretical contribution is given in Section V, where we describe our approach for AMoD by casting it into TAP, and develop its mathematical properties. In Section VI we demonstrate the power of our approach and test its scalability on realistic inputs. We conclude the paper with a discussion and future work in Section VII.

II. Related Work

There exist several approaches to study AMoD systems, spanning from simulation models [16, 22, 24] and queuing-theoretical models [55, 18] to network-flow models [30, 35, 46]. On the algorithmic side, the *control* of AMoD systems has been mostly based on network flow models employed in a receding-horizon fashion [19, 47, 48], and thresholded approximations of congestion effects [35], also accounting for the interaction with public transit [36, 37] and the power-grid [34]. In such a framework, cars can travel through a road at free-flow speed until a fixed capacity of the road is reached. At that point, no more cars can traverse it. Such models result in optimization problems solvable with off-the-shelf linear-programming solvers—making them very well suitable for *control* purposes—but lacking accuracy when accounting for congestion phenomena, which are usually described with *volume-delay functions* providing a relationship between traffic flow and travel time. In particular, the Bureau of Public Roads (BPR) developed the most commonly used volume-delay function [9], which has been applied to problems ranging from dynamic estimation of congestion [32] to route planning in agent-based models [5, 25]. Against this backdrop, a piecewise-affine approximation of the BPR function is presented in [38] and combined with convex relaxation techniques to devise a congestion-aware routing scheme for AMoD systems resulting in a quadratic program. Nevertheless, in large urban environments with several thousand transportation requests such approaches usually lead to computational times of the order of minutes, possibly rendering them less suitable for real-time control purposes.

Mathematically, AMoD can be viewed as an extension of TAP, where the latter ignores the cost and impact of rebalancing empty vehicles. Historically, TAP was introduced to model and quantify the impact of independent route choices of human drivers on themselves, and the system as a whole (see [28, 41]). Algorithmic approaches for TAP typically assume a *static* setting in which travel patterns do not change with time, allowing to cast the problem into a multi-commodity minimum-cost flow [3], which can then be formulated as a *convex programming* problem. One of the most popular tools for convex programming in the context of TAP is the Frank-Wolfe algorithm [13]. What makes this algorithm particularly suitable for solving TAP is that its direction-finding step corresponds to multiple shortest-path queries (we expand on this point in Section IV). Recent advances in pathfinding tailored for transportation networks [6], including contraction hierarchies [12, 14], have made the Frank-Wolfe approach remarkably powerful. In particular, a recent work [8] introduced a number of improvements for pathfinding, and combines those with the Frank-Wolfe method. Notably, the authors present experimental results for TAP, where their approach computes within seconds a routing scheme for up to 3 million requests over a network of a large metropolitan area. Nevertheless, such an approach is not directly applicable to AMoD problems as it would not account for the rebalancing of empty vechicles.

Finally, we mention that AMoD is closely related to *Multi-Robot Motion Planning* (MRMP), which consists of computing collision-free paths for a group of physical robots, moving them from the initial set of positions to their destination. MRMP has been studied for the setting of discrete [40, 52, 54] and continuous [11, 42, 44] domains, respectively. The unlabeled variant of MRMP [2, 45, 49, 50, 53], which involves also target assignment, is reminiscent of the rebalancing empty vehicles in AMoD, as such vehicles do not have a priori assigned destinations.

III. Preliminaries

In this section we provide a formal definition of TAP and AMoD, as our work will exploit the tight mathematical relation between these two problems.

The road network is modeled as a directed graph $G = (V, E)$: Each vertex $v \in V$ represents either a physical road intersection or points of interest on the road. Each edge $(i, j) \in E$ represents a road segment connecting two vertices $i, j \in V$. To model the travel times along the network, every edge $(i, j) \in E$ is associated with a cost function $c_{ij} : \mathbb{R}_+ \to \mathbb{R}_+$, which represents the travel time along the edge as a function of flow (i.e., traffic) along the edge per unit of time, denoted by $x_{ij} \geqslant 0$. In order to accurately capture the cost, every edge (i, j) has two additional attributes: the capacity of the edge $\kappa_{ij} > 0$ per unit of time, which can be viewed as the amount of flow beyond which the travel time will increase rapidly, and the free-flow travel time $\phi_{ij} > 0$, i.e., the *nominal* travel time when $x_{ij} = 0$. We mention that those attributes are standard when modeling traffic (see, e.g., [28]).

The time-invariant nature of this model captures the *average* value of the flows for a certain time period.

To compute c_{ij} we use the BPR function [9], which is the most widely used volume delay function. We do note that our approach presented below can be adapted to work with other functions such as the modified Davidson cost [4], so long as they induce a convex and continuously differential objective function (see Equation 1 and Section IV). Specifically, we define the cost function as

$$c_{ij}(x_{ij}) = \text{BPR}(x_{ij}, \kappa_{ij}, \phi_{ij}) \coloneqq \phi_{ij} \cdot \left(1 + \alpha \cdot \left(\frac{x_{ij}}{\kappa_{ij}}\right)^{\beta}\right),$$

where typically $\alpha = 0.15$ and $\beta = 4$.

Travel demand is represented by passenger requests $OD = \{(\lambda_m, o_m, d_m)\}_{m=1}^{M}$, where $\lambda_m > 0$ represents the amount of customers willing to travel from the origin node $o_m \in V$ to the destination node $d_m \in V$ per time unit.

A. Traffic Assignment

Here we provide a mathematical formulation of traffic assignment. We denote by $x_{ijm} \in \mathbb{R}_+$ the flow induced by request $m \in M$ on edge $(i,j) \in E$. We introduce the following constraint which ensures that the amount of flow associated with each request is maintained when a flow enters and leaves a given vertex. The amount of flow corresponding to the request $m \in M$, leaving o_m and entering d_m must match the demand flow λ_m as

$$\sum_{j \in V_i^+} x_{ijm} - \sum_{j \in V_i^-} x_{jim} = \lambda_{im}, \quad \forall i \in V, m \in M, \tag{1}$$

$$\text{where } \lambda_{im} \coloneqq \begin{cases} \lambda_m, & \text{if } o_m = i, \\ -\lambda_m, & \text{if } d_m = i, \\ 0, & \text{otherwise,} \end{cases}$$

and $V_i^- \coloneqq \{j | (i,j) \in E\}, V_i^+ \coloneqq \{j | (j,i) \in E\}$, denote heads and tails of edges leaving and entering $i \in V$, respectively. We also impose non-negative flows as

$$x_{ijm} \geqslant 0, \quad \forall (i,j) \in E. \tag{2}$$

The objective of TAP is specified in the following definition. Informally, the goal is to minimize the total travel time experienced by the users in the system, that is the sum of travel times for each individual request m.

Definition 1. The *traffic-assignment problem* (TAP) consists of minimizing the expression

$$F_E(\boldsymbol{x}) = \sum_{(i,j) \in E} x_{ij} c_{ij}(x_{ij}), \quad \text{subject to (1), (2),} \tag{3}$$

where $\boldsymbol{x} \coloneqq \left\{x_{ij} = \sum_{m \in M} x_{ijm} | (i,j) \in E\right\}$.

Observe that the cost (3) is unit-less, but is equivalent to the average travel time experienced by each user (to see this, divide the cost by the total number of requests).

B. Autonomous Mobility-on-Demand

In an AMoD system, the formulation of TAP captures only partially the cost of operating the full system. In particular, vehicles need to perform two types of tasks: (i) *occupied* vehicles drive passengers from their origins to their destinations; (ii) after dropping passengers off at their destination, *empty* vehicles need to drive to the next origin nodes, where passengers will be picked up. Indeed, the formulation of TAP above only captures the cost associated with (i), but not (ii). Another crucial difference between TAP and AMoD, which makes the latter significantly more challenging, is the fact that the travel destinations of empty vehicles are not given a priori and should be computed by the algorithm. Thus, we extend the model to include also rebalancing empty vehicles and define x_{ijr} as the rebalancing flow of empty vehicles over $(i,j) \in E$. We force empty vehicles to be rebalanced from destination nodes to origin nodes as

$$\sum_{j \in V_i^+} x_{ijr} - \sum_{j \in V_i^-} x_{jir} = r_i, \quad \forall i \in V, \tag{4}$$

$$\text{for } r_i \coloneqq \sum_{m \in M} \left(\mathbb{1}\{d_m = i\} - \mathbb{1}\{o_m = i\}\right) \lambda_m,$$

where $\mathbb{1}\{\cdot\}$ is a boolean indicator function. Observe that nodes with more arriving than departing passengers do not require rebalancing. We use $R \coloneqq \sum_{i \in V} \mathbb{1}\{r_i > 0\} r_i$ to denote the total number of rebalancing requests and enforce non-negativity of rebalancing flows as

$$x_{ijr} \geqslant 0, \quad \forall (i,j) \in E. \tag{5}$$

Definition 2. The *autonomous-mobility-on-demand problem* (AMoD) consists of minimizing the expression $F_E(\hat{\boldsymbol{x}})$, subject to (1), (2), (4), (5), where $\hat{\boldsymbol{x}} \coloneqq \{\hat{x}_{ij} \coloneqq x_{ij} + x_{ijr}\}_{(i,j) \in E}$.

C. Discussion

A few comments are in order. First, we make the assumption that mobility requests do not change in time. This assumption is justified in cities where transportation requests change slowly with respect to the average travel time [27]. Second, the model describes vehicle routes as fractional flows and it does not account for the stochastic nature of the trip requests and exogenous traffic. Given the mesoscopic perspective of our study, such an approximation is in order. Moreover, given the computational effectiveness of the approach, our algorithm is readily implementable in real-time in a receding horizon fashion, whereby randomized sampling algorithms can be adopted to compute integer-valued solutions with near-optimality guarantees [33]. Third, we assume exogenous traffic to follow habitual routes and neglect the impact of our decisions on the traffic base load, leaving the inclusion of reactive flow patterns to future work. Fourth, we model the impact of road traffic on travel time with the BPR function [9], which is well established and, despite it does not account for microscopic traffic phenomena such as traffic lights, serves the purpose of route-planning on the mesoscopic level. Finally, we constrain the capacity of the vehicles to one single customer, which is in line with current trends, and leave the extension to ride-sharing to future research [10, 48].

IV. Convex Optimization for TAP

In this section we describe the Frank-Wolfe method for convex optimization, which will later be used for solving AMoD. First, we have the following statement concerning the convexity of TAP.

Claim 1 (Convexity)**.** TAP *(Definition 1) is a convex problem.*

Proof: Given a specific edge $(i,j) \in E$, observe that the derivative of the expression $x_{ij}c_{ij}(x_{ij})$ is strictly increasing, which implies that it is convex. As the expression $F_E(\boldsymbol{x})$ consists of a sum of convex functions, it is convex as well. ■

A. The Frank-Wolfe Algorithm

Due to the convexity of the problems introduced, we can leverage convex optimization to solve TAP, and consequently AMoD, as we will see later on. Specifically, we use the Frank-Wolfe algorithm (FrankWolfe), which is well suited to our setting and has achieved impressive practical results for large-scale instances of TAP in a recent work [8].

Before introducing FrankWolfe, it should be noted that it is typically employed to minimize the *user-equilibrium* cost function captured by

$$\bar{F}_E(\boldsymbol{x}) = \sum_{(i,j)\in E} \int_0^{x_{ij}} \bar{c}_{ij}(s)\,\mathrm{d}s, \tag{6}$$

for some $\bar{c}_{ij} : \mathbb{R}_+ \to \mathbb{R}_+$, whereas we are interested in computing the *system optimum* corresponding to the minimum of $F_E(\boldsymbol{x}) = \sum_{(i,j)\in E} x_{ij}c_{ij}(x_{ij})$, using c_{ij} as defined in the previous section. However, we can enforce the user-equilibrium reached by selfish agents to correspond to the system optimum by using the *marginal costs*

$$\bar{c}_{ij}(x_{ij}) = \frac{\mathrm{d}}{\mathrm{d}x_{ij}}\left(x_{ij}c_{ij}(x_{ij})\right) = c_{ij}(x_{ij}) + x_{ij}c'_{ij}(x_{ij}),$$

which quantifies the sensitivity of the total cost with respect of small changes in flows. Specifically, to compute the system optimum, we only need to apply FrankWolfe to minimize $\bar{F}_E(\boldsymbol{x})$ as defined in (6). (See more information on this transformation in [28].) The algorithm below will be presented with respect to $\bar{F}_E$.

The following pseudo-code (Algorithm 1) presents a simplified version of FrankWolfe, which is based on [28, Chapter 4.1]. The algorithm begins with an initial solution $\boldsymbol{x}^0$, which satisfies (1), (2). To obtain $\boldsymbol{x}_0$, one can, for instance, assign each request (λ_m, o_m, d_m) to the shortest route over the traffic-free graph G, while ignoring the flows of the other users.

Algorithm 1 FrankWolfe $(\bar{F}_E, G, OD)$

1: $\boldsymbol{x}^0 \leftarrow$ feasible solution for TAP; $k \leftarrow 0$
2: **while** stopping criterion not reached **do**
3: $\quad \boldsymbol{y}^k \leftarrow \operatorname{argmin}_{\boldsymbol{y}} \bar{F}_E(\boldsymbol{x}^k) + \nabla\bar{F}_E(\boldsymbol{x}^k)^T(\boldsymbol{y}-\boldsymbol{x}^k)$, s.t. $\boldsymbol{y}^k$ satisfies (1), (2)
4: $\quad \alpha_k \leftarrow \operatorname{argmin}_{\alpha\in[0,1]} \bar{F}_E(\boldsymbol{x}^k + \alpha(\boldsymbol{y}^k - \boldsymbol{x}^k))$
5: $\quad \boldsymbol{x}^{k+1} \leftarrow \boldsymbol{x}^k + \alpha_k(\boldsymbol{y}^k - \boldsymbol{x}^k)$; $k \leftarrow k+1$
6: **return** $\boldsymbol{x}^k$

In each iteration k of the algorithm, the following steps are performed. In line 3 a value of $\boldsymbol{y}^k$ minimizing the expression $\bar{F}_E(\boldsymbol{x}^k) + \nabla\bar{F}_E(\boldsymbol{x}^k)^T(\boldsymbol{y}^k - \boldsymbol{x}^k)$, which satisfies (1), (2), is obtained. It should be noted that this corresponds to solving a linear program with respect to $\boldsymbol{y}^k$, as $\boldsymbol{x}^k$ is already known, and one is working with the gradient of $\bar{F}_E$ rather than the function itself. We will say a few more words about this computation below. In line 4 a scalar $\alpha_k \in [0,1]$ is found, such that $\bar{F}_E\left(\boldsymbol{x}^k + \alpha_k(\boldsymbol{y}^k - \boldsymbol{x}^k)\right)$ is minimized, which corresponds to solving a single-variable optimization problem, which can be done efficiently. At the end of the iteration in line 5 the solution is updated to be a linear interpolation between $\boldsymbol{x}^k$ and $\boldsymbol{y}^k - \boldsymbol{x}^k$. The last value of $\boldsymbol{x}^k$ computed before the stopping criteria has been reached, is returned in the end. Due to the convexity of the problem, it is guaranteed that as $k \to \infty$, $\boldsymbol{x}^k$ converges to the optimal solution of TAP.

B. All-or-nothing Assignment

What makes FrankWolfe particularly suitable for solving TAP is the special structure of the task of computing $\boldsymbol{y}^k$ which minimizes $\bar{F}_E(\boldsymbol{x}^k) + \nabla\bar{F}_E(\boldsymbol{x}^k)^T(\boldsymbol{y}^k - \boldsymbol{x}^k)$. First, observe that it is equivalent to minimizing the expression $\nabla\bar{F}_E(\boldsymbol{x}^k)^T\boldsymbol{y}^k$. Next, notice that for any $(i,j) \in E$ it holds that $\frac{\partial}{\partial x_{ij}}\bar{F}_E(\boldsymbol{x}^k) = \bar{c}_{ij}(x^k_{ij})$, where x^k_{ij} is the value corresponding to (i,j) of $\boldsymbol{x}^k$. That is, every variable y^k_{ij} is multiplied by $\bar{c}_{ij}(x^k_{ij})$. Thus, minimizing the expression $\nabla\bar{F}_E(\boldsymbol{x}^k)^T\boldsymbol{y}^k$ while satisfying (1), (2) is equivalent to independently assigning the shortest route for every request (λ_m, o_m, d_m), over the graph G, where the cost of traversing the edge (i,j) is independent of the traffic passing through it, and is equal to $(\nabla\bar{F}_E(\boldsymbol{x}^k))_{ij}$.

This operation is known as All-or-Nothing assignment, as each request is assigned to one specific route. Its pseudo code is given below (Algorithm 2). The ShortestPath routine returns a vector $\boldsymbol{y}^k_m$, where for every $(i,j) \in E$ that is found on the shortest path from o_m to d_m on G, weighted by $\nabla\bar{F}_E(\boldsymbol{x}^k)$, $\boldsymbol{y}^k_{m,ij} = 1$, and $\boldsymbol{y}^k_{m,ij} = 0$ otherwise.

Algorithm 2 AllOrNothing $(G, \nabla\bar{F}_E(\boldsymbol{x}^k), OD)$

1: **for** $m \in M$ **do**
2: $\quad \boldsymbol{y}^k_m \leftarrow$ ShortestPath$(G, \nabla\bar{F}_E(\boldsymbol{x}^k), o_m, d_m)$
3: **return** $\boldsymbol{y}^k := \sum_{m\in M} \lambda_m \boldsymbol{y}^k_m$

V. AMoD as TAP

In this section we establish an equivalence between TAP and AMoD. In particular, we show that a given AMoD problem can be transformed into a TAP, such that a solution to the latter, which is obtained by FrankWolfe, yields a solution to the former.

The crucial difference between the two problems is that in TAP every vehicle has a specific origin and destination vertex, whereas in AMoD this is not the case. In particular, while in AMoD empty rebalancers originate in specific destination vertices of user requests, the destinations of these rebalancers can in theory be any of the origin vertices. However, we show that this gap can be bridged by supplementing the original graph G with an additional "dummy" vertex, and connecting

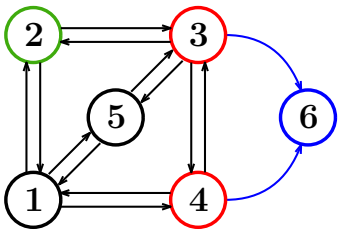

Fig. 2. A simple example for the construction. The graph G consists of the vertices 1 to 5 and the (black) edges between them. The demand for G is $OD = \{(2,1,2),(1,2,4),(1,3,4),(2,4,1),(2,4,2)\}$, where each triplet denotes the intensity, origin, and destination, respectively. Red vertices indicate shortage of incoming vehicles (e.g., four passengers depart from vertex 4, but only two vehicles arrive), green indicates excess (e.g, four vehicles terminate in vertex 2 but only two passengers leave), black represents vertices with met demand (e.g., vertex 1 where two vehicles terminating, and two passengers departing). Consequently, the graph G' additionally contains the dummy vertex 6, and edges $(3,6),(4,6)$ drawn in blue, originating from vertices of G with shortage. Accordingly, the capacity is set to $\kappa_{3,6}=1, \kappa_{4,6}=2$. OD' extends OD with the request $(3,2,6)$. Observe that its intensity corresponds to the total excess in vertices $3,4$.

to it edges emanating from all the vertices that need to be rebalanced. We then set the costs of the edges to guarantee an almost complete rebalancing, i.e., only a small fraction of the rebalancing requests will not be fulfilled. In the remainder of this section we provide a detailed description of the approach and proceed to analyze its theoretical guarantees.

A. The construction

We formally describe the structure of this new graph $G' = (V', E')$, where $V' = V \cup \{n\}$, and $E' = E \cup \{(i,n) | i \in V \text{ and } r_i < 0\}$. Recall that $r_i < 0$ indicates that there are fewer user requests arriving to $i \in V$ than there are departing from the vertex, which implies that rebalancers should be sent to this vertex. The vertex $n \notin V$ is new and will serve as dummy target vertex for all the rebalancers. See example in Figure 2.

To ensure that a sufficient number of rebalancers will arrive at each vertex that needs to be rebalanced, we assign to every edge (i,n) the cost $c_{in}(x_{in}) = \text{BPR}(x_{in}, \kappa_{in}, \phi_{in})$, where $\kappa_{in} = -r_i$, and $\phi_{in} = L$, where L is a large constant whose value will be determined later on.

The final ingredient in transforming AMoD to TAP is providing excess vehicles with specific origins and destinations. Given the original set of requests OD, for every $i \in V$ such that $r_i > 0$ we add the request (r_i, o_i, n), where r_i is its intensity. This yields the extended requests set OD'.

As each free or occupied vehicle has a specific destination, we can think of the AMoD problem as a new TAP problem over the graph G' and the extended set of requests OD'. As rebalancers are no longer needed to be considered separately from the users, we may redefine x_{ij} to be the total flow along an edge $(i,j) \in E'$, including users and rebalancers. Denote $\bar{E} \coloneqq E' \setminus E$. The cost function for the corresponding TAP is $F_{E'}(\boldsymbol{x}) = F_E(\boldsymbol{x}) + F_{\bar{E}}(\boldsymbol{x})$, where $F_{\bar{E}}(\boldsymbol{x}) = \sum_{(i,n)\in\bar{E}} x_{in} c_{in}(x_{in})$.

We show in the remainder of this section that after choosing L correctly, then $\boldsymbol{x}^*$, which minimizes the system optimum $F_{E'}(\boldsymbol{x}^*)$ under the constraints (1), (2), with respect to G', OD', represents a high-quality solution to the AMoD problem, in which the majority of rebalancing requests are fulfilled. It is worth clarifying that the cost of the obtained flow is represented by $F_E(\boldsymbol{x})$.

B. Analysis

First, we note that the new objective function $F_{E'}$ remains convex owing to the fact that for every new dummy edge its cost function is monotone and increasing with respect to its flow (see Claim 1). Given a vector assignment $\boldsymbol{x}$ for TAP over G', it will be useful to split it into variables $\boldsymbol{x}_E$ corresponding to the edges E, and variables $\boldsymbol{x}_{\bar{E}}$ corresponding to $\bar{E}$.

The motivation for setting the specific capacity value κ_{in} to edge $(i,n) \in \bar{E}$ is given in the following lemma. Recall that $R \coloneqq \sum_{i\in V} \mathbb{1}\{r_i > 0\} r_i$.

Lemma 1 (Optimal assignment for dummy edges)**.** *Let $\boldsymbol{x}^*$ minimize $F_{\bar{E}}(\boldsymbol{x}^*)$, under the constraint that $\sum_{(i,n)\in\bar{E}} x^*_{in} = R$. Then $\boldsymbol{x}^*_{\bar{E}} = \boldsymbol{\kappa}$, where $\boldsymbol{\kappa} = \{\kappa_{in} | (i,n) \in \bar{E}\}$.*

Proof: Note that $F_{\bar{E}}(\boldsymbol{x}_{\bar{E}})$ is convex, and feasible for the constraint $\sum_{x_{in}\in\boldsymbol{x}_{\bar{E}}} x_{in} = R$. Thus, it has a unique minimum. To find it, we shall use Lagrange multipliers.

Let $g(\boldsymbol{x}) \coloneqq \sum_{j=1}^{m} x_j$ and define the Lagrangian $\mathcal{L}(\boldsymbol{x}, \lambda) \coloneqq F_{\bar{E}}(\boldsymbol{x}) - \lambda(g(\boldsymbol{x}) - R)$. For any $x_j \in \boldsymbol{x}$,

$$\frac{\partial}{\partial x_j}\mathcal{L} = L\left(1 + 0.75\left(\frac{x_j}{\kappa_j}\right)^4\right) - \lambda, \text{ and } \frac{\partial}{\partial\lambda}\mathcal{L} = R - g(\boldsymbol{x}).$$

As $\frac{\partial}{\partial x_j}\mathcal{L} = 0$, $x_j = \kappa_j \left(\frac{\lambda}{0.75L} - \frac{1}{0.75}\right)^{1/4}$, which is then plugged into $\frac{\partial}{\partial\lambda}\mathcal{L} = 0$, where we use the fact that $\sum_{j=1}^{\ell} \kappa_j = R$, to yield $\lambda = 1.75L$. We then substitute λ to yield $x_j = \kappa_j$, which concludes the proof. ∎

We arrive at the main theoretical contribution of the paper, which states that L can be tuned to obtain a solution where the fraction of unfulfilled rebalancing requests $\delta > 0$ is as small as desired. Notice that for a given solution $\boldsymbol{x}$, the expression $\frac{\|\boldsymbol{x}_{\bar{E}} - \boldsymbol{\kappa}\|_1}{2R}$ represents the fraction of unfulfilled requests.

Theorem 1 (Bounded fraction of unfulfilled requests)**.** *Let $\boldsymbol{x}^* \coloneqq \text{argmin}_{\boldsymbol{x}} F_{E'}(\boldsymbol{x})$ subject to (1), (2). For every $\delta \in (0,1]$ exists $L_\delta \in (0,\infty)$ such that if $L > L_\delta$ then $\frac{\|\boldsymbol{x}^*_{\bar{E}} - \boldsymbol{\kappa}\|_1}{2R} \leqslant \delta$.*

Proof: Let $\boldsymbol{x}^0$ be an assignment satisfying (1), (2) such that $\boldsymbol{x}^0_{\bar{E}} = \boldsymbol{\kappa}$, which minimizes the expression $F_{E'}(\boldsymbol{x}^0)$. That is, $\boldsymbol{x}^0$ minimizes $F_{E'}$ over all $\boldsymbol{x}$ which fully satisfies the rebalancing constraints. (Observe that such $\boldsymbol{x}^0$ represents the optimal solution of the original AMoD problem.) If $\boldsymbol{x}^0$ turns out to yield the minimum of $F_{E'}(\boldsymbol{x})$ without conditioning on $\boldsymbol{x}^0_{\bar{E}} = \boldsymbol{\kappa}$ then the result follows. Thus, we assume otherwise. Fix $\delta \in (0,1]$ and let $\boldsymbol{x}^\delta$ represent any assignment, satisfying (1), (2), and for which it holds that $\frac{\|\boldsymbol{x}^\delta_{\bar{E}} - \boldsymbol{\kappa}\|_1}{2R} > \delta$.

Our goal is to find a value of L_δ such that for any $L > L_\delta$ it holds that $F_{E'}(\boldsymbol{x}^\delta) > F_{E'}(\boldsymbol{x}^0)$. This implies that using such L we are guaranteed that if a solution returned by FrankWolfe will have at most δ unfulfilled request. This is equivalent to requiring that

$$F_{\bar{E}}(\boldsymbol{x}^\delta) - F_{\bar{E}}(\boldsymbol{x}^0) > F_E(\boldsymbol{x}^0) - F_E(\boldsymbol{x}^\delta).$$

Notice that by Lemma 1, it holds that $F_{\bar{E}}(\boldsymbol{x}^\delta) - F_{\bar{E}}(\boldsymbol{x}^0) > 0$. Recovering a precise upper bound for the expression $F_E(\boldsymbol{x}^0) - F_E(\boldsymbol{x}^\delta)$, which depends on δ, is quite difficult.

We therefore resort to a crude (over-)estimation of it, which is the value $F_E(\boldsymbol{x}^0)$. Thus, we wish to find L such that $F_{\bar{E}}(\boldsymbol{x}^\delta) - F_{\bar{E}}(\boldsymbol{x}^0) > F_E(\boldsymbol{x}^0)$.

Define $\Delta_{in} := x_{in} - \kappa_{in}$, for every $(i,n) \in \bar{E}$, where $x_{in} \in \boldsymbol{x}^\delta_{\bar{E}}$. For every such x_{in} the following holds:

$$\begin{aligned} c_{in}(x_{in}) &= L\left(1 + 0.15\left(\frac{x_{in}}{\kappa_{in}}\right)^4\right) \\ &\geqslant L\left(1 + 0.15\left(1 + \frac{4\Delta_{in}}{\kappa_{in}}\right)\right) = L\left(1.15 + \frac{0.6\Delta_{in}}{\kappa_{in}}\right), \end{aligned}$$

where the inequality follows from Bernoulli's inequality. Also, note that $c_{in}(\kappa_{in}) = L \cdot 1.15$. Thus,

$$\begin{aligned} F_{\bar{E}}(\boldsymbol{x}^\delta) &= \sum_{(i,n)} x_{in} c_{in}(x_{in}) \geqslant \sum_{(i,n)} (\kappa_{in} + \Delta_{in}) L\left(1.15 + \frac{0.6\Delta_{in}}{\kappa_{in}}\right) \\ &= \sum_{(i,n)} L\left(1.15 \cdot \kappa_{in} + 1.75 \cdot \Delta_{in} + \frac{0.6\Delta_{in}^2}{\kappa_{in}}\right) \\ &= \sum_{(i,n)} 1.15 L\kappa_{in} + \sum_{(i,n)} 1.75 L\Delta_{in} + \sum_{(i,n)} L\frac{0.6\Delta_{in}^2}{\kappa_{in}} \\ &= F_{\bar{E}}(\boldsymbol{x}^0) + 0 + \sum_{(i,n)} L\frac{0.6\Delta_{in}^2}{\kappa_{in}} \geqslant F_{\bar{E}}(\boldsymbol{x}^0) + \frac{0.6L}{R}\sum_{(i,n)} \Delta_{in}^2 \\ &\geqslant F_{\bar{E}}(\boldsymbol{x}^0) + \frac{0.6L}{R} \cdot \ell^{-1}\left(\sum_{(i,n)} |\Delta_{in}|\right)^2 \\ &= F_{\bar{E}}(\boldsymbol{x}^0) + \frac{0.6L}{R\ell}\left\|\boldsymbol{x}^\delta_{\bar{E}} - \boldsymbol{x}^0_{\bar{E}}\right\|_1^2 \geqslant F_{\bar{E}}(\boldsymbol{x}^0) + 2.4RL\ell^{-1}\delta^2, \end{aligned}$$

where the second to last inequality is due to Cauchy-Schwarz, and ℓ is the number of dummy edges.

We have just shown that for any $\boldsymbol{x}^\delta$ it holds that $F_{\bar{E}}(\boldsymbol{x}^\delta) - F_{\bar{E}}(\boldsymbol{x}^0) \geqslant 2.4RL\ell^{-1}\delta^2$. To conclude, for $L = \frac{F_E(\boldsymbol{x}^0)}{2.4R\ell^{-1}\delta^2}$, it follows that $F_{E'}(\boldsymbol{z}^\delta) > F_{E'}(\boldsymbol{z}^0)$, which implies that any value $\boldsymbol{x}^*$ satisfying the constraints (1), (2) and minimizing $F_{E'}(\boldsymbol{x})$, also guarantees that $\frac{\|\boldsymbol{x}^*_{\bar{E}} - \boldsymbol{\kappa}\|_1}{2R} \leqslant \delta$. ∎

We will consider the practical aspects of computing a proper L in Section VI. The next corollary is the final piece of the puzzle. It proves that when using a proper L, not only that δ is bounded, but also the value $F_E(\boldsymbol{x}^*)$ is at most $F_E(\boldsymbol{x}^0)$.

Corollary 1 (Bounded cost of routing)**.** *Fix $\delta \in (0,1]$ and let $L \in (0,\infty)$ such that $L > L_\delta$. Then (i) $\frac{\|\boldsymbol{x}^*_{\bar{E}} - \boldsymbol{\kappa}\|_1}{2R} \leqslant \delta$, and (ii) $F_E(\boldsymbol{x}^*) < F_E(\boldsymbol{x}^0)$, where $\boldsymbol{x}^* = \operatorname{argmin}_{\boldsymbol{x}} F_{E'}(\boldsymbol{x})$ under constraints (1), (2).*

Proof: Let $\boldsymbol{x}^*, \boldsymbol{x}^0$ be as defined in the previous proof. It is clear that x^* satisfies (i). Now, observe that by definition $F_{E'}(\boldsymbol{x}^*) < F_{E'}(\boldsymbol{x}^0)$, and by Lemma 1, $F_{\bar{E}}(\boldsymbol{x}^0) < F_{\bar{E}}(\boldsymbol{x}^*)$. Then the following derivation proves (ii): $F_E(\boldsymbol{x}^*) + F_{\bar{E}}(\boldsymbol{x}^*) < F_E(\boldsymbol{x}^0) + F_{\bar{E}}(\boldsymbol{x}^0) < F_E(\boldsymbol{x}^0) + F_{\bar{E}}(\boldsymbol{x}^*)$. ∎

VI. Experimental results

In this section, we use experimental results to demonstrate the power of our approach for AMoD routing via a reduction to TAP (Section V) on a real-world case study. In the first set of experiments in Section VI-C we validate experimentally the theory developed in Section V-B. In summary, we observe that the approach yields near-optimal solutions within a few seconds, where most (more than 99%) of the rebalancing requests are fulfilled, when L is properly tuned. We then test the scalability of the approach on scenarios involving as much as 600k user requests, where we observe that running times and convergence rates scale only linearly with the size of input. In the final set of experiments we demonstrate the benefit of the approach over previous methods.

A. Implementation Details

All results were obtained using a commodity laptop equipped with 2.80GHz × 4 core i7-7600U CPU, and 16GB of RAM, running 64bit Ubuntu 18.04 OS. The C++ implementation of the FrankWolfe algorithm is adapted (with merely minor changes) from the `routing-framework`, which was developed for [8]. For shortest-path computation in the AllOrNothing routine, we use *contraction hierarchies* [14], which are embedded in the `routing-framework`, and are in turn based upon the code in the `RoutingKit` (see [12]). Running times reported below are for a 4-core parallelization.

In our experiment we observed that in some situations, the first few iterations of FrankWolfe route most of the rebalancers to a select set of dummy edges, so that the actual flow is far larger than the capacity. This leads to numeric overflows when working with the standard C++ `double` and to failure of the program. We mention that this phenomenon was not observed for the modified Davidson cost function [4], linearized at 95% of the capacity, which has a more gentle gradient. To alleviate the problem when working with BPR, one can resort to using `long double` or linearize the value of the function after a certain threshold is reached. We chose the latter, by linearizing at 500% of the capacity. We emphasize that this does not affect the final outcome of the algorithm.

Finally, we mention that we experimented with a few cost functions (including linear, and modified Davidson) for the dummy edges, until we settled on BPR, which yields the best convergence rates.

B. Data

Similarly to [38], our experiments are conducted over Manhattan in New York City, where the OD-pairs are inferred from taxi rides that occurred during the morning peak hour on March 1st, 2012. As in [38], we assume to centrally control all ride-hailing vehicles in Manhattan, and accordingly scale taxi rides requests by a factor of 6. The total number of real user OD-pairs in our experiments is thus $6 \times 25{,}960$ (unless stated otherwise). In order to take into consideration the fact that autonomous vehicles need to share the road with private vehicles, which should increase the overall cost of travelling along edges, we introduce a parameter of exogenous traffic (as was done in [38]). In particular, for a non-dummy edge $(i,j) \in E$, with a flow x_{ij}, we assign the cost $c_{ij}\left(x_{ij} + x^e_{ij}\right)$, where x^e_{ij} denotes the exogenous flow. For simplicity, we set this value so that the fraction of x^e_{ij} over the capacity κ_{ij} of the edge κ_{ij} is the same, over all edges. That is, we choose a value $\gamma_{\text{exo}} \geqslant 0$ and set $x^e_{ij}/\kappa_{ij} = \gamma_{\text{exo}}$. Unless stated otherwise, $\gamma_{\text{exo}} = 0.8$, which approximates the scenario of the rush-hour traffic, mentioned above. The underlying road-map $G = (V,E)$ was extracted from an Open Street Map [15], where $|V| = 1352, |E| = 3338$.

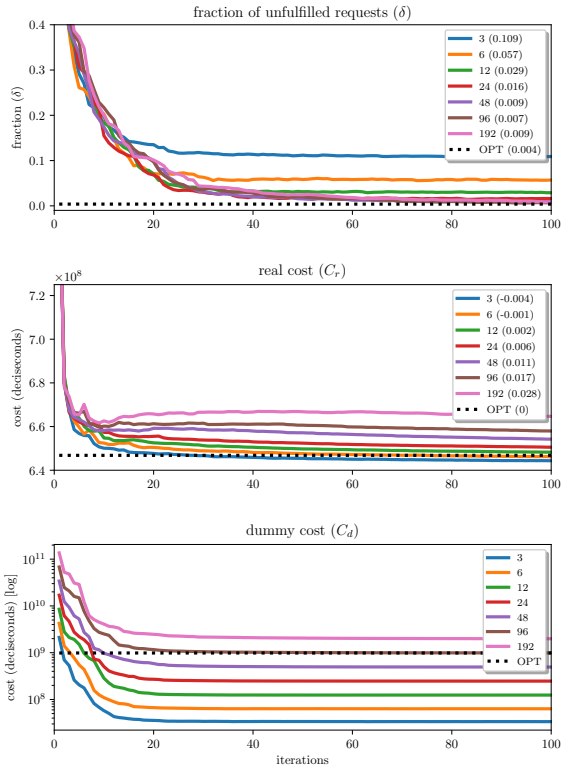

Fig. 3. Validation of theoretical results. For all the plots, OPT represents the corresponding optimal value for $L = 96$. [TOP] A plot of δ. The left number in the legend represents L, whereas the value in the brackets denotes δ after iteration 100. For instance, OPT yields $\delta = 0.004$, whereas for the same L after 100 iterations we have that $\delta = 0.007$. [CENTER] A plot of C_r. The left number in the legend represents L, whereas the value in brackets denotes $(C_r - \text{OPT})/\text{OPT}$ after iteration 100. E.g., C_r for $L = 96$ is only 1.7% larger than OPT. [BOTTOM] A plot for C_d.

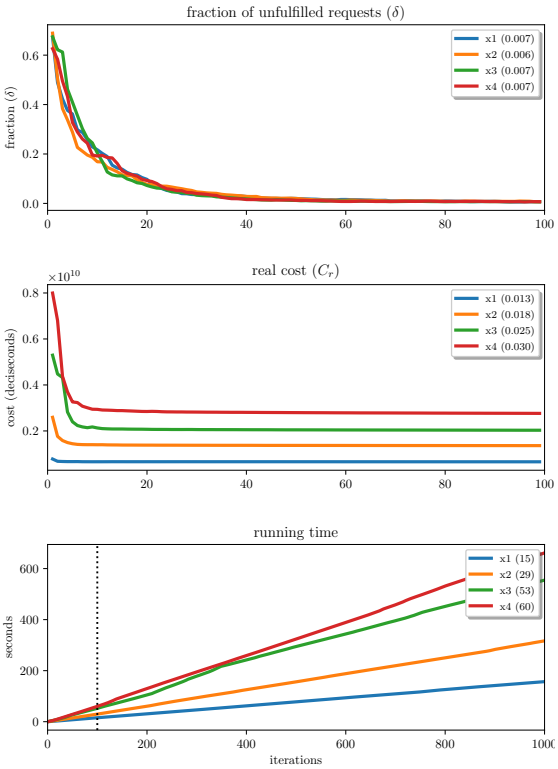

Fig. 4. Plots for scalability experiments. [TOP] A plot of δ, i.e., fraction of unfulfilled rebalancing requests. The left notation in the legend ($\times i$) indicates the used number of copies of the original OD set, whereas the value in brackets denotes δ after iteration 100. [CENTER] A plot of C_r. The value in brackets denotes the ratio of deviation for C_r between the iterations 100 and 1000. [BOTTOM] A plot of total running time. The value in brackets denotes the running time after iteration 100.

C. Results

Before proceeding to the experiments we mention that in some results we compare the outcome of our algorithm with an optimal value, denoted by OPT. To obtain it, we run the algorithm, with a corresponding set of parameters, for 10,000 iterations. This provides a good approximation of the real optimum. E.g., for the final iteration of the algorithm, when $L = 96$min, the relative difference in the real and dummy cost is $1.64 \cdot 10^{-7}$ and $2.91 \cdot 10^{-7}$, respectively. The terms real and dummy costs correspond to F_E and $F_{\bar{E}}$, respectively.

Validation of the theory. Our first set of experiments is designed to validate the convergence of the approach, and the theory presented in Section V. In summary, we observe that with already small L a solution where a large majority of rebalancing requests are fulfilled is achieved. Moreover, the system cost for the rebalaced system is also very close to the optimal value. Importantly, this is achieved within only 100 iterations of FRANKWOLFE, corresponding to around 15s.

In order to test how the value of L affects the fraction of unbalanced requests δ, as stated in Theorem 1, and the real and dummy costs of solution, i.e., $C_r := F_E(\boldsymbol{x})$, $C_d := F_{\bar{E}}(\boldsymbol{x})$, respectively, we experiment with values of L in the range of 3 to 192min (see Figure 3).

In terms of the fraction of unfulfilled requests δ, as Theorem 1 states, increasing L reduces this value. For instance, when $L = 3$, $\delta = 0.109$, but already when $L = 48$ then $\delta = 0.009$, after 100 iterations. It should be noted that a small value of δ is reached only when C_d is noticeably larger than C_r (see middle and bottom plots in Figure 3, for comparison).

This implies that our estimation for L suggested in Theorem 1 is quite conservative. From a practical point of view, we recommend iterating over L using binary search until a desired value of δ is achieved.

We now discuss the relation between the magnitude of L and C_r. We observe that C_r increases with L. This follows from the fact that a smaller δ requires rebalancers to increase the total length of their trips in order to accommodate more requests. Here we can also observe a possible drawback in setting L to be needlessly large, as it takes more iterations to settle on the correct value of C_r. This follows from the fact that to obtain a smaller δ requires more iterations.

Lastly, observe that already after approximately 50 iterations, all three values reach a plateau, i.e., increasing the number of iterations only slightly changes the corresponding value. Also note that values of OPT, computed for $L = 96$, are very close to corresponding values for the same L after only 100 iterations. This indicates that a small number of iterations suffices to reach a near-optimal value, be it δ, C_r, or C_d.

Scalability. In this set of experiments we demonstrate the scalability of the approach by increasing the total number of OD requests. We use the set of $6 \times 25{,}960$ OD pairs, which was utilized in the above experiments, as a basis, and then multiply this set by $1, 2, 3$ and 4. The last case includes 623,040 travel

requests. In an attempt to make the four settings similar in terms of the total traffic on the road, we pair each setting with exogenous traffic of $\gamma_{\text{exo}} \in \{0.8, 0.6, 0.4, 0.2\}$, respectively.

Each scenario was executed for 1000 iterations. For the first two plots we chose to show results only for the first 100 iterations, as the change is very minor from this point on. We also omit a plot for the dummy cost as we found it uninformative. The results are presented in Figure 4.

The convergence rate and the final result for δ, C_r behave similarly for the four settings. However, we note that we expect to see a more significant difference for a more realistic data set, as a bigger data set would have more diverse OD pairs. Nevertheless, we emphasize that our four settings do not yield similar solutions with respect to flow distribution, as the scale of flow affects the cost. In terms of the running time, we mention that it scales proportionally to the number of OD pairs, and the rate of change with respect to the number of iterations is roughly constant.

Comparison with previous work. We demonstrate the benefits of computing a solution to AMoD using the precise formulation of the cost function BPR, as opposed to a piecewise-affine approximation of BPR, or a congestion-unaware solution. The piecewise-affine approximation approach was suggested in [38], where the BPR function is approximated by two affine functions (see more details in Section II). In terms of computation time, our approach yields the results in around 15s, whereas the reported running times of [38] are around 4min for similar hardware. We wish to point out that [38] do not minimize travel time for rebalancing vehicles, and also include walking times in their analysis. The congestion-unaware approach, which was utilized in earlier works (see, e.g., [29]), generates a solution without considering the effect of congestion of the routed vehicles on the overall cost.

We implemented all three types of solution using our framework, by replacing the precise formulation of the BPR function, where relevant. We wish to clarify that the cost function used on the dummy edges remains BPR, to guarantee rebalancing (see Section V-B), as this does not affect the real solution cost. The computation was done with $L = 96$. The same applies to OPT which was computed using the full BPR.

We ran the three approaches for varying values of $\gamma_{\text{exo}} \in [0, 2]$ (see plots of the comparison in Figure 5). As was observed in previous studies, congestion-unaware planning underestimates the real travel cost, which results in plans that divert traffic to overly congested routes. The deviation from OPT increases with exogenous traffic. Already for $\gamma_{\text{exo}} = 0.8$, the total cost is around 1.3 times OPT. Approximate BPR is much more accurate in this respect. However, it either under- or overestimates the real cost, which yields plans where vehicles are rerouted from low-cost routes to more congested routes. Approximate BPR yields plans whose deviation from OPT is twice as high for the precise BPR, when $\gamma_{\text{exo}} \in [0, 1.1]$; it coincides with our solution for $\gamma_{\text{exo}} = 1.2$; for larger values of $\gamma_{\text{exo}} \in [0, 1.1]$ the deviation becomes more noticeable. For instance, when $\gamma_{\text{exo}} = 1.3$ it yields a solution of around 1.1 times OPT. In contrast, our approach yields an accurate estimation of OPT for the entire range of γ_{exo}.

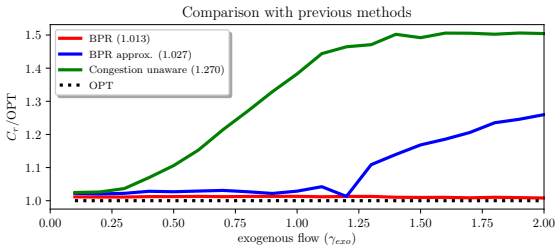

Fig. 5. Plots for comparison experiments. For each of the three types of cost functions we plot the ratio between the obtained cost C_r and the cost for OPT, as a function γ_{exo} (x-axis). The corresponding value for $\gamma_{\text{exo}} = 0.8$ is given in the brackets.

VII. Discussion and Future Work

This paper presented a computationally-efficient framework to compute the system-optimal operation of a fleet of self-driving cars providing on-demand mobility in a congestion-aware fashion. To the best of our knowledge, this is the first scheme providing high-quality routing solutions for large-scale AMoD fleets within seconds, thus enabling real-time implementations for operational control through receding horizon optimization (to account for new information that is revealed over time) and randomized sampling (to recover integer flows with near-optimality guarantees [33]). Our approach consists of reducing our problem to a TAP instance, such that an optimal solution achieved for the latter yields an optimal for AMoD. This allows us to leverage modern and highly effective approaches for TAP. We showcased the benefits of our approach in a real-world case-study. The results showed our method outperforming state-of-the-art approaches by at least one order of magnitude in terms of computational time, whilst improving the system performance by up to 20%.

This work opens the field for several extensions, and leaves a few open questions. On the side of theory, our immediate goal is to analyze the convergence rate of our approach. There is an abundance of recent results on the theoretical properties of FrankWolfe (see, e.g., [21, 31]), which regained popularity in recent years due to applications in machine learning [20]. However, it is not clear whether these results are applicable to our setting, and what their implications are on the convergence of the real cost F_E, rather than $F_{E'}$. We also plan to investigate approaches to obtain a more informative estimation of the constant L (see Theorem 1). On the implementation side, we mention that the performance can be further improved by using customizable contraction hierarchies [12] for pathfinding, and bundling identical OD pairs. We also point out the fact that the AllOrNothing routine is embarrassingly parallel, and a significant speedup can be gained from using a multi-core machine. Finally, on the application side, we aim at exploiting the high computational efficiency of the presented approach by implementing it in real-time using receding horizon schemes. To this end, it might be necessary to employ a time-expansion of the road-graph and account for stochastic effects, as it was done in [18, 47]. In addition, it is of interest to extend this framework to capture the interaction with public transit [36] and the power grid [34], and account for the interaction of self-driving vehicles with the urban infrastructure.

Acknowledgments

We would like to thank Valentin Buchhold for advising on the `routing-framework`. We thank Dr. Ilse New and Michal Kleinbort for proofreading this paper and providing us with their comments and advice. Our special thanks are also extended to our labmates Ramón Darío Iglesias, Matthew Tsao, and Jannik Zgraggen for the fruitful discussions. The second author would like to express his gratitude to Dr. Chris Onder for his support. This research was supported by the National Science Foundation under CAREER Award CMMI-1454737 and the Toyota Research Institute (TRI). The first author is also supported by the Fulbright Scholars Program. This article solely reflects the opinions and conclusions of its authors and not NSF, TRI, Fulbright, or any other entity.

References

[1] The world factbook. Central Intelligence Agency, 2018. Available at https://www.cia.gov/library/publications/the-world-factbook/fields/2212.html.

[2] A. Adler, M. De Berg, D. Halperin, and K. Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Trans. Automation Science and Engineering*, 12(4):1309–1317, 2015.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Pearson, 1993.

[4] R Akcelik. A new look at Davidson's travel time function. *Traffic Engineering & Control*, 19(N10), 1978.

[5] J. Aslam, S. Lim, and D. Rus. Congestion-aware traffic routing system using sensor data. In *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, 2012.

[6] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. *Route Planning in Transportation Networks*, pages 19–80. Springer International Publishing, 2016.

[7] P. Berger. Mta blames uber for decline in new york city subway, bus ridership. The Wall Street Journal, 2018. available online.

[8] V. Buchhold, P. Sanders, and D. Wagner. Real-time traffic assignment using fast queries in customizable contraction hierarchies. In *International Symposium on Experimental Algorithms*, pages 27:1–27:15, 2018. URL https://github.com/vbuchhold/routing-framework.

[9] Bureau of Public Roads. Traffic assignment manual. Technical report, U.S. Dept. of Commerce, Urban Planning Division, 1964.

[10] M. Cáp and J. Alonso-Mora. Multi-objective analysis of ridesharing in automated mobility-on-demand. In *Robotics: Science and Systems*, 2018.

[11] M. Cáp, Novák P., A. Kleiner, and M. Selecký. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Trans. Automation Science and Engineering*, 12(3):835–849, 2015.

[12] J. Dibbelt, B. Strasser, and D. Wagner. Customizable contraction hierarchies. *ACM Journal of Experimental Algorithmics*, 21(1):1.5:1–1.5:49, 2016. URL https://github.com/RoutingKit/RoutingKit.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[14] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

[15] M. Haklay and P. Weber. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 7 (4):12–18, 2008.

[16] S. Hörl, C. Ruch, F. Becker, E. Frazzoli, and K. W. Axhausen. Fleet control algorithms for automated mobility: A simulation assessment for Zurich. In *Annual Meeting of the Transportation Research Board*, 2018.

[17] W. Hu. Your Uber car creates congestion. should you pay a fee to ride? The New York Times, 2017. available online.

[18] R. Iglesias, F. Rossi, R. Zhang, and M. Pavone. A BCMP network approach to modeling and controlling Autonomous Mobility-on-Demand systems. In *Workshop on Algorithmic Foundations of Robotics*, 2016.

[19] R. Iglesias, F. Rossi, K. Wang, D. Hallac, J. Leskovec, and M. Pavone. Data-driven model predictive control of autonomous mobility-on-demand systems. In *Proc. IEEE Conf. on Robotics and Automation*, 2018.

[20] Martin Jaggi and Zaid Harchaoui. ICML 2014 Tutorial on Frank-Wolfe and Greedy Optimization for Learning with Big Data, 2017. https://sites.google.com/site/frankwolfegreedytutorial, Last accessed on 2019-05-09.

[21] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *Neural Information Processing Systems*, pages 496–504, 2015.

[22] M. W. Levin, K. M. Kockelman, S. D. Boyles, and T. Li. A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application. *Computers, Environment and Urban Systems*, 64:373 – 383, 2017.

[23] J. I. Levy, J. J. Buonocore, and K. Von Stackelberg. Evaluation of the public health impacts of traffic congestion: a health risk assessment. *Environmental Health*, 9(1):65, 2010.

[24] M. Maciejewski, J. Bischoff, S. Hörl, and K. Nagel. Towards a testbed for dynamic vehicle routing algorithms. In *Int. Conf. on Practical Applications of Agents and Multi-Agent Systems - Workshop on the application of agents to passenger transport (PAAMS-TAAPS)*, 2017.

[25] E. Manley, T. Cheng, A. Penn, and A. Emmonds. A framework for simulating large-scale complex urban traffic dynamics through hybrid agent-based modelling. *Computers, Environment and Urban Systems*, 44:27–36, 2014.

[26] R. Molla. Americans seem to like ride-sharing services like Uber and Lyft. But it's hard to say exactly how many use them. Recode, 2018. Available at https://www.recode.net/2018/6/24/17493338/ride-sharing-services-uber-lyft-how-many-people-use.

[27] H. Neuburger. The economics of heavily congested roads. *Transportation Research*, 5(4):283–293, 1971.

[28] Michael Patriksson. *The Traffic Assignment Problem: Models and Methods*. Dover Publications, 2015. ISBN 0486787907.

[29] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus. Load balancing for Mobility-on-Demand systems. In *Robotics: Science and Systems*, 2011.

[30] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus. Robotic load balancing for Mobility-on-Demand systems. *Int. Journal of Robotics Research*, 31(7):839–854, 2012.

[31] J. Peña, D. Rodríguez, and N. Soheili. On the von Neumann and Frank-Wolfe algorithms with away steps. *SIAM Journal on Optimization*, 26(1):499–512, 2016.

[32] A. Rivas, G. Inmaculada, S. Sánchez-Cambronero, R. M. Barba, and L. Ruiz-Ripoll. A continuous dynamic traffic assignment model from plate scanning technique. *Transport Research Procedia*, 18:332–340, 2016.

[33] F. Rossi. *On the Interaction between Autonomous Mobility-on-Demand Systems and the Built Environment: Models and Large Scale Coordination Algorithms*. PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, 2018.

[34] F. Rossi, R. Iglesias, M. Alizadeh, and M. Pavone. On the interaction between Autonomous Mobility-on-Demand systems and the power network: Models and coordination algorithms. In *Robotics: Science and Systems*, 2018. Extended version available at https://arxiv.org/abs/1709.04906.

[35] F. Rossi, R. Zhang, Y. Hindy, and M. Pavone. Routing autonomous vehicles in congested transportation networks: Structural properties and coordination algorithms. *Autonomous Robots*, 42(7):1427–1442, 2018.

[36] M. Salazar, F. Rossi, M. Schiffer, C. H. Onder, and M. Pavone. On the interaction between autonomous mobility-on-demand and the public transportation systems. In *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, 2018. In Press. Extended Version, Available at https://arxiv.org/abs/1804.11278.

[37] M. Salazar, N. Lanzetti, F. Rossi, M. Schiffer, and M. Pavone. Intermodal autonomous mobility-on-demand. *IEEE Transactions on Intelligent Transportation Systems*, 2019. URL ../wp-content/papercite-data/pdf/Salazar.ea.T-ITS19.pdf. Submitted.

[38] 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 Conference*, 2019.

[39] D. Schrank, T. Lomax, and S. Turner. The 2007 urban mobility report. Technical report, Texas Transportation Institute, 2007.

[40] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[41] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice Hall, 1985.

[42] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris. dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots*, 2019.

[43] F. Siddiqui. Failing transit ridership poses an 'emergency' for cities, experts fear. The Washington Post, 2018. available online.

[44] K. Solovey, J. Yu, O. Zamir, and D. Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems*, 2015.

[45] Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *I. J. Robotics Res.*, 35(14):1750–1759, 2016.

[46] K. Spieser, K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone. Toward a systematic approach to the design and evaluation of Autonomous Mobility-on-Demand systems: A case study in Singapore. In *Road Vehicle Automation*. Springer, 2014.

[47] M. Tsao, R. Iglesias, and M. Pavone. Stochastic model predictive control for autonomous mobility on demand. In *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, 2018. In Press. Extended Version, Available at https://arxiv.org/pdf/1804.11074.

[48] M. Tsao, D. Milojevic, C. Ruch, M. Salazar, E. Frazzoli, and M. Pavone. Model predictive control of ride-sharing autonomous mobility on demand systems. In *Proc. IEEE Conf. on Robotics and Automation*, 2019. In Press.

[49] M. Turpin, N. Michael, and V. Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *I. J. Robotics Res.*, 33(1):98–112, 2014.

[50] M. Turpin, K. Mohta, N. Michael, and V. Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Auton. Robots*, 37(4):401–415, 2014.

[51] B. Tuttle and T. Cowles. Traffic jams cost americans $124 billion in 2013. *Time - Money*, 2014.

[52] J. Yu. Constant factor time optimal multi-robot routing on high-dimensional grids. In *Robotics: Science and Systems*, 2018.

[53] J. Yu and S. M. LaValle. Distance optimal formation control on graphs with a tight convergence time guarantee. In *IEEE Conference on Decision and Control*, pages 4023–4028, 2012.

[54] J. Yu and S. M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Trans. Robotics*, 32(5):1163–1177, 2016.

[55] R. Zhang and M. Pavone. Control of robotic Mobility-on-Demand systems: A queueing-theoretical perspective. *Int. Journal of Robotics Research*, 35(1–3):186–203, 2016.