2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Quantification on the Efficiency Gain of Automated Ridesharing Services

Shuo Han, Ufuk Topcu, George J. Pappas

Abstract— **Ridesharing services often require constant rebalancing vehicle supply in order to meet passenger demand across the transportation network. We compare the cost of rebalancing between two different methods for controlling the vehicle flows: (1) direct control, which models on-demand dispatchable vehicles such as autonomous vehicles and (2) indirect control based on price differences, which models human drivers as in the current ridesharing scheme. We propose a metric that quantifies the efficiency gain of automated ridesharing (i.e., direct control) based on the maximum difference between the rebalancing cost of two methods. The benefit of the proposed metric is that it is independent of the actual demand and only relies on properties of the transportation network. We present a set of numerical tools for computing the metric. For a general graph, the metric can be computed using an efficient local search method called the difference-of-convex algorithm (DCA). Numerical experiments on a practical graph (constructed from a pricing map for the Washington, DC area) show that the DCA often converges within a few iterations. For fully connected and symmetric graphs, the metric can be computed from an equivalent convex program. Moreover, the convex program adopts a simple closed-form optimal solution.**

I. Introduction

The growing urbanization in recent years has brought significant stress on modern urban transportation systems. Aside from building new infrastructures, it is also imperative to fully exploit the capacity of current transportation systems. In recent years, on-demand ridesharing services (e.g., Uber and Lyft) have gained increasing popularity, especially at places where owning a car is expensive (such as major cities) or public transportation is underdeveloped.

One key to successful operation of on-demand ridesharing services is to maintain a reasonable wait time for potential passengers. To achieve this, it is necessary to relocate available vehicles to places with high passenger demand. Without any guidance, human drivers who provide ridesharing services are not able to respond to changes in passenger demand. To facilitate the rebalancing of vehicles, the current method adopted by ridesharing companies is setting non-uniform prices for different regions in the city: if a region has high passenger demand, then passengers originated from that region need to pay more expensive fares. Since the price changes are also instantaneously visible to drivers, available drivers are motivated to relocate to regions with high demand.

On the other hand, we have witnessed a growing trend of bringing autonomous vehicles into urban transportation, in which ridesharing service providers are one of the major advocates. In August 2016, Uber launched a test program in Pittsburgh, PA where customers can call autonomous cars for their Uber rides [4]. Compared to price-based control, autonomous vehicles can be dispatched on demand and are expected to have more flexibility in rebalancing. In previous work, researchers have proposed control strategies for balancing autonomous/dispatchable vehicles using a number of different approaches such as receding-horizon control and queueing-theoretic methods [7], [9], [12].

Up to now, however, there is still a lack of tools for quantifying the efficiency gained from autonomous vehicles from a systemic point of view. Such tools will be useful for providing guidelines on various aspects in system design such as optimal fleet size and spatial distribution. In this paper, we quantify the efficiency by the cost of rebalancing, which is measured by the total amount of vehicle flow required to reach the balanced condition (i.e., same supply-demand mismatch for all regions).

The response of drivers to regional prices are modeled using a proportional law, under which the flow of drivers between two regions is proportional to the price difference between the regions. Under the proportional law, the vehicle flows are analogous to electric current flows in a resistor network, and the regional prices are analogous to nodal voltages. Specifically, the flows are related to solutions of a Laplacian linear system, which is a linear system of equations with a graph Laplacian as the coefficient matrix. Previous work, mostly motivated by problems in power systems, has investigated important properties of Laplacian flows and how flows vary with changes in graph topology (e.g., removal of an edge) [6], [10], [11]. In comparison, our work focuses on comparing the difference between unconstrained flows and Laplacian flows on a given fixed network.

Contribution: In this paper, we propose a metric that quantifies the efficiency gain of automated ridesharing (i.e., direct control) compared with price-based control based on the maximum difference between the rebalancing cost. The benefit of the proposed metric is that it is independent of the actual demand and only relies on properties of the transportation network. We present a set of numerical tools for computing the metric. For a general graph, we show that the metric can be computed using an efficient local search method called the difference-of-convex algorithm

S.H. and G.J.P. are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104. {hanshuo,pappasg}@seas.upenn.edu. U.T. is with the Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Austin, TX 78712. utopcu@utexas.edu. This work was supported in part by the NSF (CNS-1239224, CNS-1239152), US Department of Transportation through the UTC program, and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

(DCA). Through numerical experiments on a practical graph (constructed from a pricing map for the Washington, DC area), we show that the DCA often converges extremely quickly (within a few iterations). For the special case of fully connected and symmetric graphs, we derive an equivalent convex program for computing the metric. We also prove that the convex program adopts a simple closed-form optimal solution.

Paper organization: The paper is organized as follows. Section II presents a graph-theoretic model for price-based control on a ridesharing network and introduces the metric (i.e., maximum gap) used for quantifying the efficiency gain from direct control. Section III discusses properties of the metric for general networks and a numerical method named DCA for computing the metric. Section IV focuses on a special case where the graph is fully connected and symmetric and shows how the metric can be computed using convex optimization. Section V presents simulations for both the case of fully connected and symmetric graphs and that of general graphs; for general graphs, we use a graph constructed from a realistic pricing map from Uber for the Washington, DC area.

II. Problem Formulation

A. System model

We consider an area of interest (e.g., a city) consisting of n geographical regions and model the entire area as an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ represents the geographical regions. Throughout the paper, we may use the term *regions* and *vertices* interchangeably to refer to the set V. We say that two regions i and j are *connected* if and only if $(i, j) \in E$. The edges E are used to model the flows of vehicles across the regions. In our model, it is not necessary for two connected regions to be geographically adjacent. For any two regions i and j, we have $(i, j) \in E$ as long as drivers are willing to move between these two regions. As a rule of thumb, we assume that drivers are willing to move to a different region if the driving time is less than a certain threshold (e.g., 20 minutes).

We denote by $\delta_i \in \mathbb{R}$ the *mismatch* between supply and demand in region i. As a convention, when $\delta_i > 0$, it implies that there is more demand (passengers) than supply (drivers) in region i. We call $\delta = (\delta_1, \delta_2, \ldots, \delta_n) \in \mathbb{R}^n$ the *mismatch profile* for the entire area.

For any two connected regions i and j, we denote by $f_{ij} \in \mathbb{R}$ the *flow of supply* from i to j, and we always have

$$f_{ij} = -f_{ji}, \quad \forall (i, j) \in E. \tag{1}$$

We consider two different methods for controlling the flow: direct control and indirect control. In the case of *direct control*, the flows can be assigned directly. This is the case when the supply consists of autonomous vehicles that can be dispatched on demand. On the other hand, in the case of *indirect control*, the flows are controlled by prices. This is the case when vehicles are operated by independent human drivers and models how current on-demand ridesharing services (e.g., Uber and Lyft) are operated. Specifically, for indirect control, region i can set its own *price of service* $p_i \in \mathbb{R}$ for which passengers need to pay when requesting services within the region. Because of the incentives induced by price differences among regions, drivers are motivated to relocate to regions with higher prices. We assume that the flow f_{ij} is proportional to the price difference between regions i and j as follows:

$$f_{ij} = -\alpha_{ij}(p_i - p_j), \tag{2}$$

where $\alpha_{ij} > 0$ is called the *sensitivity* of drivers to the price difference between regions i and j. When $p_i > p_j$, the model (2) implies $f_{ij} < 0$, which captures the fact that drivers tend to move to regions with higher prices. In practice, the sensitivity α_{ij} is related to the effort (e.g., driving time) that drivers need to make in order to move between regions i and j. We denote by $A = [\alpha_{ij}] \in \mathbb{R}_+^{n \times n}$ the matrix of sensitivity. For notational convenience, we let $\alpha_{ij} = 0$ for all $(i, j) \notin E$.

The flows of supply will alter the supply-demand mismatch profile δ. In our model, we consider the case in which, throughout the rebalancing phase, (1) no new demand is generated and (2) no vehicle leaves the area. For any given initial mismatch profile δ and flows of supply f, the new mismatch profile δ' can be obtained as follows:

$$\delta'_i = \delta_i + \sum_{j=1}^{n} f_{ij}, \quad \forall i \in V. \tag{3}$$

From (1) and (3), we can derive a conservation law for δ:

$$\sum_{i=1}^{n} \delta'_i = \sum_{i=1}^{n} \delta_i. \tag{4}$$

Throughout the paper, we assume that the graph G is connected, i.e., for any two vertices $i, j \in V$, there exists a path connecting i and j.

Assumption 1. *The graph G is connected.*

This assumption is reasonable for most practical transportation networks of interest: a vehicle should be able to move to any regions within the area of interest.

B. Cost of control

Our main goal of controlling the flow is to balance the supply-demand mismatch across the entire area. We say that a mismatch profile $\delta \in \mathbb{R}^n$ is *balanced* if and only if δ satisfies $\delta_i = \delta_j$ for any $i, j \in V$. From the conservation law (4), we know that for a given initial mismatch profile δ, the final balanced profile δ' satisfies

$$\delta'_i = \bar{\delta} \triangleq \frac{1}{n} \sum_{i=1}^{n} \delta_i. \tag{5}$$

We quantify the cost of control using the total amount of flow used for achieving the balanced profile (5). The *minimum cost of indirect control* $c_I(\delta)$ is defined as the

optimal value of the following optimization problem:

$$c_I(\delta) := \min_{p,f} \quad \frac{1}{2}\sum_{i,j} |f_{ij}| \tag{6}$$
$$\text{s.t.} \quad \sum_{j=1}^{n} f_{ij} = \bar{\delta} - \delta_i, \quad \forall i$$
$$f_{ij} = -\alpha_{ij}(p_i - p_j), \quad \forall i, j.$$

The factor $\frac{1}{2}$ in the objective function is used to account for the fact $|f_{ij}| = |f_{ji}|$.

Similar to the definition (6) for c_I, the *minimum cost of direct control* $c_D(\delta)$ is defined as the optimal value of the following optimization problem:

$$c_D(\delta) := \min_{g} \quad \frac{1}{2}\sum_{i,j} |g_{ij}| \tag{7}$$
$$\text{s.t.} \quad \sum_{j=1}^{n} g_{ij} = \bar{\delta} - \delta_i, \quad \forall i$$
$$g_{ij} = -g_{ji}, \quad \forall i, j.$$

In (7), the flows g do not need to satisfy the rule (2) and are treated as decision variables, whereas in (6) the flows f are slack variables controlled by the prices p.

Note that both c_I and c_D depend on the graph G, but we often drop the dependence on G when it is clear from the context. Both problems (6) and (7) are convex optimization problems (in fact, linear programs). As a result, there are efficient numerical algorithms for computing $c_I(\delta)$ and $c_D(\delta)$ for any given δ.

C. Maximum gap in cost

From the definitions of c_I and c_D, we can immediately see that $c_I(\delta) \geq c_D(\delta)$ for any δ, because any feasible solution to the optimization problem (6) is also feasible for problem (7). For a given graph G, the gap in cost $c_I(\delta) - c_D(\delta)$ may vary with δ, as shown in the following example.

Example 2. Consider the fully connected graph (Fig. 1) with $n = 3$ and $\alpha_{ij} = 1$ for all $(i, j) \in E$. When $\delta = (2, 1, 0)$, we can obtain the costs $c_I(\delta)$ and $c_D(\delta)$ as follows:

$$c_I(\delta) = \frac{4}{3}, \text{ achieved at } f_{12} = -\frac{1}{3},\ f_{23} = -\frac{1}{3},\ f_{31} = \frac{2}{3}$$
$$\text{(with } p_1 = \frac{2}{3},\ p_2 = \frac{1}{3},\ p_3 = 0),$$
$$c_D(\delta) = 1, \text{ achieved at } g_{12} = 0,\ g_{23} = 0,\ g_{31} = 1,$$

so that the gap $c_I(\delta) - c_D(\delta) = \frac{1}{3}$.

When $\delta = (1, 0, 0)$, however, we can obtain the costs $c_I(\delta)$ and $c_D(\delta)$ as follows:

$$c_I(\delta) = \frac{2}{3}, \text{ achieved at } f_{12} = -\frac{1}{3},\ f_{23} = 0,\ f_{31} = \frac{1}{3}$$
$$\text{(with } p_1 = \frac{1}{3},\ p_2 = 0,\ p_3 = 0),$$
$$c_D(\delta) = \frac{2}{3}, \text{ achieved at } g_{12} = -\frac{1}{3},\ g_{23} = 0,\ g_{31} = \frac{1}{3},$$

so that the gap $c_I(\delta) - c_D(\delta) = 0$.

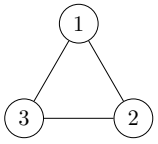


Fig. 1. Graph used in Example 2 ($\alpha_{ij} = 1$ for all i, j).

For analyzing the efficiency gain from direct control, it is natural to seek a metric that is independent of δ and relies only on the graph G. This is because G only depends on the configuration of road networks, whereas δ depends on the particular instantiation of passenger demand and hence is difficult to obtain *a priori*. In this paper, we choose the metric to be the *maximum gap* in cost, defined as

$$\gamma \triangleq \max_{\delta \in \Delta} [c_I(\delta) - c_D(\delta)] \tag{8}$$

for some uncertainty set Δ. The choice of Δ depends on the estimation for the supply-demand mismatch in the area of interest. One suitable choice of Δ is a box defined by $\Delta = \{\delta \colon \delta_L \preceq \delta \preceq \delta_U\}$ for some $\delta_L, \delta_U \in \mathbb{R}^n$, where $\preceq$ indicates entry-wise inequality. Our problem to study in this paper is to compute γ for any given graph G and Δ.

Problem 3. For any given graph G and uncertainty set Δ, compute the maximum gap γ.

In the next few sections, we will first comment on the challenges of computing γ for a general graph. Then, we will show that there is an efficient local search algorithm that computes γ for general graphs when Δ is convex. In the end, we discuss the special case of fully connected and symmetric graphs, which allows γ to be computed from a convex optimization problem.

III. Maximum Gap: General Case

A. Feasibility of rebalancing

Before presenting methods for computing the maximum gap γ, we would like to show that the optimization problems (6) and (7) are feasible for any δ. As we have discussed in Section II-C, the set of feasible solutions for problem (6) is contained in that of problem (7). As a result, we only need to show that problem (6) is feasible for any δ.

Define $q \in \mathbb{R}^n$ such that $q_i \triangleq \delta_i - \bar{\delta}$. We can write the constraints of problem (6) as

$$\sum_{j=1}^{n} \alpha_{ij}(p_i - p_j) = q_i$$

or more compactly as

$$Lp = q,$$

where $L = \mathrm{diag}(A\mathbf{1}) - A$. Here, we have used the notation $\mathrm{diag}(v)$ to denote the $n \times n$ diagonal matrix with elements of $v \in \mathbb{R}^n$ on the diagonal and $\mathbf{1}$ to denote the vector of all ones. Observe that L is the (weighted) Laplacian matrix associated with G by recalling the definition of graph Laplacian: for

any weighted graph G with weighted adjacency matrix $A = [\alpha_{ij}]$, the graph Laplacian L is an $n \times n$ symmetric matrix with

$$L_{ij} = \begin{cases} \sum_{j=1}^{n} \alpha_{ij} & i = j, \\ -\alpha_{ij} & i \neq j. \end{cases}$$

By using properties of the Laplacian matrix, we can not only show that problem (6) is feasible for any δ, but also that the problem adopts a unique solution for the flows f.

Proposition 4. *For any $\delta \in \mathbb{R}^n$, the problem (6) of rebalancing with indirect control is feasible. Additionally, the optimal flow f in problem (6) is unique.*

Proof: In order to show that problem (6) is feasible, we only need to show that the linear system of equations $Lp = q = \delta - \bar{\delta}\mathbf{1}$ always has a solution. It is known that $\lambda_1 = 0$ is always an eigenvalue of L. Moreover, when G is connected (Assumption 1), the eigenvalue $\lambda_1 = 0$ has multiplicity of 1, and the corresponding eigenvector is $v_1 = \mathbf{1}$ [5]. In other words, the null space of L is spanned by $v_1 = \mathbf{1}$. As a result, if $y \in \mathbb{R}^n$ lies in the range space of L, then we have $\mathbf{1}^T y = 0$. It is not difficult to verify that q satisfies $\mathbf{1}^T q = \mathbf{1}^T \delta - \bar{\delta}\mathbf{1}^T\mathbf{1} = 0$, which implies that q lies in the range space of L. In other words, the system $Lp = q$ always has a solution.

Because the null space of L is spanned by $v_1 = \mathbf{1}$, if both p and p' are solutions for $Lp = q$, we must have $p = p' + a\mathbf{1}$ for some constant $a \in \mathbb{R}$. From the relationship (2), we can see that both p and p' will lead to the same flow f. Namely, there is only one feasible solution for the flow f in problem (6), so that the unique solution f is the optimal flow for problem (6). ∎

From Proposition 4, instead of defining c_I through the optimization problem (6), we can rewrite c_I as

$$c_I(\delta) = \frac{1}{2} \sum_{i,j} |\alpha_{ij}(p_i - p_j)|, \quad Lp = \delta - \bar{\delta}\mathbf{1}.$$

B. A numerical method for computing the maximum gap

Before discussing numerical methods for computing the maximum gap γ, we would like to show that both functions c_I and c_D are convex.

Proposition 5. *Both functions c_I and c_D are convex.*

Proof: The function c_I is the sum of absolute values of linear functions and hence is convex. For the function c_D, notice that the Lagrange dual problem of (7) can be obtained as

$$\begin{aligned} \max_{\nu,\mu,\lambda^+,\lambda^-} \quad & \frac{1}{2} \sum_{i=1}^{n} \nu_i(\bar{\delta} - \delta_i) \\ \text{s.t.} \quad & 1 - \lambda_{ij}^+ - \lambda_{ij}^- = 0, \quad \forall i, j \\ & -\nu_i + \mu_{ij} + \mu_{ji} + \lambda_{ij}^+ - \lambda_{ij}^- = 0, \quad \forall i, j. \end{aligned} \tag{9}$$

Since problem (7) is a linear program and always feasible (as a result of Proposition 4), we know that strong duality holds in this case [3], so that we can redefine c_D through the dual problem (9). Then, the convexity of c_D follows from the fact that c_D is a pointwise maximum of affine functions in δ as implied from the dual problem (9). ∎

As a result from Proposition 5, the problem (8) of computing γ requires maximizing the difference $c_I - c_D$ of two convex functions or, equivalently, minimizing the difference $c_D - c_I$ of two convex functions. In general, the function $c_D - c_I$ is neither convex nor concave, so that we cannot use standard convex optimization algorithms to find its global optimal solution. Instead, we need to resort to local search algorithms in order to compute γ.

When the uncertainty set Δ is convex, we can find local optimal solutions for problem (8) using the *difference-of-convex algorithm (DCA)* [1], [8]. The DCA, which is outlined in Algorithm 1, can be used for finding local optimal solutions for minimization problems whose objective function has the form $u - v$, where both u and v are convex (but not necessarily differentiable). In Algorithm 1, we use $\operatorname{dom} u$ to denote the domain of u, the symbol $\partial v(x)$ to denote the subdifferential (set of subgradients) of v at x, and u^* to denote the convex conjugate of u:

$$u^*(y) \triangleq \sup\left\{y^T x - u(x) \colon x \in \mathbb{R}^n\right\}.$$

The key step for implementing the DCA is finding subgradients of v and u^*. As it turns out, both subgradients are easy to compute for our case. In our case, we have $u = c_D$ and $v = c_I$, and the sequence of decision variables is $\{\delta_k\}_{k=1}^{K+1}$. The function c_I is the sum of absolute values of linear functions, whose subgradient can be obtained in closed form: for $w(x) = |c^T x|$, the subdifferential

$$\partial w(x) = \begin{cases} c & c^T x > 0, \\ -c & c^T x < 0, \\ \{\lambda c \colon \lambda \in [-1, 1]\} & c^T x = 0. \end{cases}$$

For finding subgradients of $u^* = c_D^*$, we need to use an important fact from convex analysis. When u is proper, lower semicontinuous, and convex, it can be shown that $x_{k+1} \in \partial u^*(y_k)$ if and only if [2]

$$x_{k+1} \in \arg\min\left\{u(x) - y_k^T x \colon x \in \mathbb{R}^n\right\}.$$

Using the definition of c_D, we see that the step of finding $\delta_{k+1} \in \partial c_D^*(y_k)$ requires computing an optimal solution for the following optimization problem (with g being treated as

Algorithm 1 Difference-of-convex algorithm (DCA).

Objective function: $u - v$ (both u and v are convex)
Input: $x_1 \in \operatorname{dom} u$, number of iterations K
Output: x_{K+1}
  for $k = 1, 2, \ldots, K$ **do**
    Find $y_k \in \partial v(x_k)$
    Find $x_{k+1} \in \partial u^*(y_k)$
  end for

a slack variable):

$$\begin{aligned}\min_{\delta,g} \quad & \frac{1}{2}\sum_{i,j}|g_{ij}| - y_k^T\delta \\ \text{s.t.} \quad & \sum_{j=1}^{n} g_{ij} = \bar{\delta} - \delta_i, \quad \forall i \\ & g_{ij} = -g_{ji}, \quad \forall i,j,\end{aligned}$$

which is a linear program with a similar computational complexity as evaluating c_D itself.

We would like to emphasize that the DCA is a local search method whose result depends on the initial condition (x_1 is Algorithm 1). As will be presented in Section V, our numerical experiments show that the DCA is quite suitable for our case. Besides the fact that the subgradients can be easily computed, the DCA often converges extremely quickly (less than 10 iterations), which allows using a large number of random initial conditions in order to search for the global optimum.

IV. Maximum Gap: Fully-Connected and Symmetric Graph

In this section, we will show that, under special circumstances, the problem of computing γ can be formulated as a convex optimization problem. From Section III, we can see that the challenge in computing γ is that the optimization problem (8) involves maximizing the difference of two convex functions or, equivalently, the sum of a convex and a concave function $c_I + (-c_D)$. Consequently, the best that we can wish for is that, for some cases, the function c_I becomes affine, which makes the objective function $c_I - c_D$ concave and turns problem (8) into a convex optimization problem.

The main result in this section is that when G is fully connected and symmetric, and the set $\Delta = \{\delta \colon \delta_i \in [-1,1]\}$ (i.e., a hypercube), the optimization problem (8) can be reformulated as a convex program.

Definition 6 (Fully connected and symmetric graph)**.** A graph $G = (V,E)$ with weight matrix $A = [\alpha_{ij}]$ is called *fully connected and symmetric* if there exists $\bar{\alpha} \in \mathbb{R}$ such that $\alpha_{ij} = \bar{\alpha}$ for all $i,j \in V$.

For the case of indirect control, we can see that $\bar{\alpha}$ does not affect the cost, because the prices p can be scaled accordingly with $\bar{\alpha}$ to achieve the same flow by the relationship (2). Without loss of generality, we use $\bar{\alpha} = 1$ in all following results.

When the graph is fully connected and symmetric, we can show that c_I adopts a simple form.

Proposition 7. *When the graph G is fully connected and symmetric, we have*

$$c_I(\delta) = \frac{1}{2n}\sum_{i,j}|\delta_i - \delta_j|.$$

Proof: When the graph G is fully connected and symmetric (with $\alpha_{ij} = 1$). For any $i,j \in V$, we have

$$\begin{aligned}-\sum_{k=1}^{n}(p_i - p_k) &= \bar{\delta} - \delta_i, \\ -\sum_{k=1}^{n}(p_j - p_k) &= \bar{\delta} - \delta_j,\end{aligned}$$

or equivalently

$$-\left[(n-1)p_i - \sum_{k\neq i} p_k\right] = \bar{\delta} - \delta_i, \tag{10}$$

$$-\left[(n-1)p_j - \sum_{k\neq j} p_k\right] = \bar{\delta} - \delta_j, \tag{11}$$

Subtracting (11) from (10), we obtain

$$n(p_i - p_j) = \delta_i - \delta_j,$$

so that

$$c_I(\delta) = \frac{1}{2}\sum_{i,j}|f_{ij}| = \frac{1}{2}\sum_{i,j}|p_i - p_j| = \frac{1}{2n}\sum_{i,j}|\delta_i - \delta_j|$$

■

Although Proposition 7 allows us to rewrite c_I in a simple form, the function is still not affine. However, because c_I now only depends on the difference $\delta_i - \delta_j$, we can write c_I as an affine function (in fact, a linear function) if we know the ordering of $\{\delta_i\}_{i=1}^n$, as shown in the following corollary.

Corollary 8. *When the graph G is fully connected and symmetric, we have*

$$c_I(\delta) = \frac{1}{n}\sum_{(i,j)\in\mathcal{I}(\delta)}(\delta_i - \delta_j), \quad \mathcal{I}(\delta) \triangleq \{(i,j)\colon \delta_i \geq \delta_j\}.$$

Recall that the set Δ under consideration is a hypercube. Because of the symmetry in both the objective function $c_I - c_D$ and the constraint set Δ, we can add an additional constraint

$$\delta_1 \leq \delta_2 \leq \cdots \leq \delta_n$$

to the optimization problem (8) without affecting its optimal value. Namely, we have

$$\max_{\delta\in\Delta}[c_I(\delta) - c_D(\delta)] = \max_{\delta\in\Delta'}[c_I(\delta) - c_D(\delta)],$$

where $\Delta' = \{\delta\colon \delta_i \in [-1,1],\ \delta_i \geq \delta_j \text{ for } i > j\}$. Using Corollary 8, we can obtain our main result.

Proposition 9. *Suppose the graph G is fully connected and symmetric. When $\Delta = \{\delta\colon \delta_i \in [-1,1]\}$, we have*

$$\gamma = \max_{\delta\in\Delta'}\left[\frac{1}{n}\sum_{i>j}(\delta_i - \delta_j) - c_D(\delta)\right], \tag{12}$$

where $\Delta' = \{\delta\colon \delta_i \in [-1,1],\ \delta_i \geq \delta_j \text{ for } i > j\}$.

Proof: When G is fully connected symmetric, it can be verified that both functions c_I (from Proposition 7) and c_D are permutation invariant. Namely, for any $\delta \in \mathbb{R}^n$,

we have $c_I(\delta) = c_I(\text{perm}(\delta))$ and $c_D(\delta) = c_D(\text{perm}(\delta))$, where $\text{perm}(\delta) \in \mathbb{R}^n$ is any vector generated by permuting the elements of δ. As a result, if δ^* is an optimal solution for problem (8), we can alway construct $\hat{\delta}^* = \text{perm}(\delta^*)$ with $\hat{\delta}_i^* \geq \hat{\delta}_j^*$ for $i > j$, which also satisfies $\hat{\delta}^* \in \Delta$ and achieves the same optimal value as δ^*. This implies that we can introduce an additional constraint

$$\delta_i \geq \delta_j \text{ for } i > j$$

to Δ without affecting the optimal value of problem (8). ■

Notice that the objective function in (12) is concave, so that problem (12) is a convex program. The immediately benefit of reformulating the original problem (8) as a convex program (12) using Proposition 9 is that we can use efficient numerical algorithms to find γ, instead of using local search methods such as the DCA for general graphs.

Remark 10. Notice that $-c_D$ is defined through a maximization problem. The problem for finding γ in Proposition 9, if written fully, becomes

$$\begin{aligned} \max_{\delta,g} \quad & \frac{1}{n}\sum_{i>j}(\delta_i - \delta_j) - \frac{1}{2}\sum_{i,j}|g_{ij}| \\ \text{s.t.} \quad & \sum_{j=1}^{n} g_{ij} = \bar{\delta} - \delta_i, \quad \forall i \\ & g_{ij} = -g_{ji}, \quad \forall i, j, \\ & \delta_i \in [-1, 1], \qquad \delta_i \geq \delta_j \text{ for } i > j. \end{aligned}$$

We would like to point out an interesting fact about the new convex program (12). It can be shown that the optimal solution of (12) can be expressed in closed form.

Proposition 11. *Suppose the graph G is fully connected and symmetric. When $\Delta = \{\delta \colon \delta_i \in [-1, 1]\}$, the following δ attains the maximum gap:*

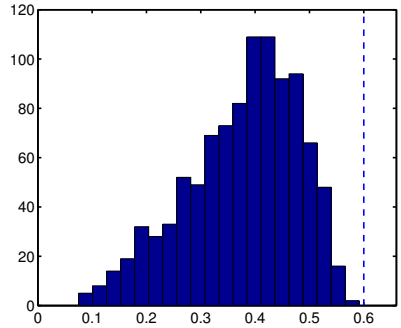
$$\delta_i = \begin{cases} -1 & 1 \leq i \leq \frac{n+2}{4}, \\ 0 & \frac{n+2}{4} < i \leq \frac{n+1}{2}, \\ -\delta_{n-i+1} & \frac{n+1}{2} < i \leq n. \end{cases}$$

Proposition 11 allows us to quickly compute γ without even having to numerically solve the optimization problem (12) (although efficient numerical methods are available). We omit the proof due to page limitation.

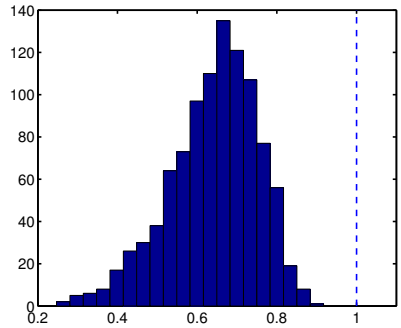
V. Numerical Results

A. Fully connected and symmetric graph

For the case of fully connected and symmetric graph (with Δ chosen as the unit hypercube), we compare the result for γ obtained by two different methods. One is through solving the convex optimization problem in Proposition 9; the other is the Monte Carlo method by evaluating the objective function $c_I - c_D$ at random samples within Δ. Fig. 2 shows the results of these two methods for two different graph sizes ($n = 5$ and $n = 8$), where the number of samples for the Monte Carlo method is 1,000. The histogram summarizes the evaluations of the cost difference at the random samples, whereas the dashed line indicates the result obtained using Proposition 9. As expected, because the result from solving the convex optimization problem in Proposition 9 is guaranteed to yield the maximum gap, the dashed line serves as an upper bound for all the evaluations in the histogram. We have also used the closed-form solution as given by Proposition 11, and the results coincide with the numerical solutions obtained from Proposition 9.



(a) $n = 5$



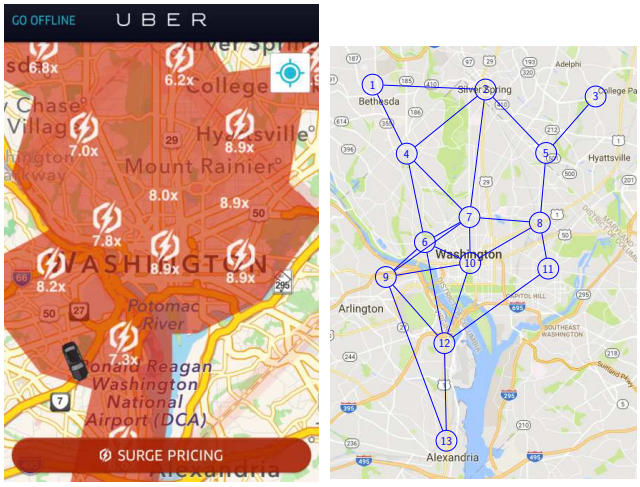
(b) $n = 8$

Fig. 2. Results of the maximum gap γ. Histogram: results from the Monte Carlo method. Dashed line: results from solving the optimization problem in Proposition 9. The set $\Delta = \{\delta \colon \delta_i \in [-1, 1]\}$.

B. General graph: A case study for Washington, DC

For the case of general graphs, we considered a realistic case for the Washington, DC metropolitan area. Fig. 3 shows the graph G used in our numerical experiments. Fig. 3a is a screenshot of the distribution of surge pricing factors obtained from the Uber app (visible to Uber drivers). Each numerical value on the map represents a surge pricing zone. When surge pricing is present, passengers need to pay the normal fare multiplied by the surge pricing factor corresponding to their location. Fig. 3b shows the graph G constructed from the 13 pricing zones in Fig. 3a. Two vertices of G were chosen to be connected if the driving time between the center of the two corresponding pricing zones is less than 20 minutes, which is what we believe the maximum time that human drivers are willing to spend on relocation. The weights of the graph were chosen to be inversely proportional to the driving time between the zone centers.

Fig. 4 shows the results of the maximum gap γ for the graph in Fig. 3b with $\Delta = \{\delta \colon \delta_i \in [-1, 1]\}$. Because



(a) (b)

Fig. 3. (a) Surge pricing map from Uber. (b) Graph G used in simulation. Each vertex represents a surge pricing zone in (a). Two vertices are connected if the driving time between the center of the two corresponding pricing zones is less than 20 minutes.

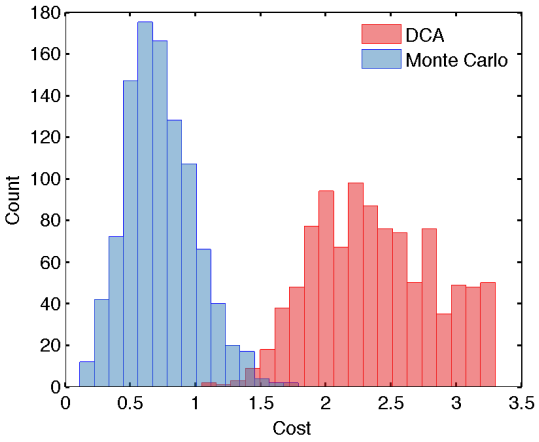


Fig. 4. Results of the maximum gap γ for the graph in Fig. 3b. The set $\Delta = \{\delta \colon \delta_i \in [-1, 1]\}$.

the graph is not fully connected and symmetric, we can no longer be guaranteed to find a global optimum for problem (8). Instead, we compare the results obtained by the DCA (Section III-B) with the Monte Carlo method. We used a total of 1,000 random samples. Each sample was used as the initial condition for the DCA, whereas the sample was directly evaluated for the Monte Carlo method. The results are summarized as histograms in Fig. 4. The maximum cost obtained by the Monte Carlo method is 1.78, whereas the one obtained by the DCA is 3.29. For the result obtained by the DCA, the maximum cost is achieved with $c_I(\delta) = 11.29$ and $c_D(\delta) = 8.00$, which implies that the use of direct control incurs a saving of about 29%.

It can be seen from the histograms that the DCA significant improves the results by the Monte Carlo method. For all random initial conditions, the DCA converged in less than 10 iterations. Although we cannot guarantee that $\gamma = 3.29$ obtained by the DCA is the global maximum, we suspect that it is close to maximum, as we tried increasing the number of random initial conditions from 1,000 to 20,000 but the maximum did not change.

VI. Conclusions

We studied the cost of rebalancing vehicle supply (for ridesharing services) on a transportation network modeled as an undirected graph. We compared two different methods for controlling the vehicle flows: (1) direct control, which models on-demand dispatchable vehicles such as autonomous vehicles and (2) indirect control based on price differences, which models human drivers as in the current ridesharing scheme. We proposed a metric that quantifies the efficiency gain of direct control (i.e., automated ridesharing) based on the maximum difference between the rebalancing cost of two methods. The benefit of the proposed metric is that it is independent of the actual demand and only relies on properties of the transportation network.

Generally, computing the metric requires solving an optimization problem. Our main result is a set of numerical tools for computing the metric. For a general graph, the metric can be computed using an efficient local search method called the difference-of-convex algorithm (DCA). Numerical experiments on a practical graph (constructed from a pricing map for the Washington, DC area) showed that the DCA often converges extremely quickly. For fully connected and symmetric graphs, the metric can be computed from an equivalent convex program. Moreover, the convex program adopts a simple closed-form optimal solution.

References

[1] L. T. H. An, M. T. Belghiti, and P. D. Tao. A new efficient algorithm based on DC programming and DCA for clustering. *Journal of Global Optimization*, 37(4):593–608, 2007.

[2] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[3] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[4] M. Chafkin. Uber's first self-driving fleet arrives in Pittsburgh this month. Bloomberg, August 2016. `http://bloom.bg/2bzThsU` (retrieved: August 19, 2016).

[5] F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[6] C. Lai and S. H. Low. The redistribution of power flow in cascading failures. In *Allerton Conference on Communication, Control, and Computing*, pages 1037–1044, 2013.

[7] F. Miao, S. Han, S. Lin, Q. Wang, J. Stankovic, A. Hendawi, D. Zhang, T. He, and G. J. Pappas. Data-driven robust taxi dispatch under demand uncertainties. *IEEE Transactions on Control Systems Technology*. (under review).

[8] N. M. Nam, D. Giles, and R. B. Rector. Minimizing differences of convex functions and applications to facility location and clustering. *arXiv preprint arXiv:1511.07595*, 2015.

[9] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus. Robotic load balancing for mobility-on-demand systems. *The International Journal of Robotics Research*, 31(7):839–854, 2012.

[10] S. Soltan, D. Mazauric, and G. Zussman. Cascading failures in power grids: analysis and algorithms. In *Proceedings of the 5th international conference on Future energy systems*, pages 195–206, 2014.

[11] A. J. Wood and B. F. Wollenberg. *Power generation, operation, and control*. John Wiley & Sons, 2012.

[12] R. Zhang and M. Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, 35(1-3):186–203, 2016.