Achieving Target Equilibria in Network Routing Games without Knowing the Latency Functions

Umang Bhaskar, Katrina Ligett, Leonard J. Schulman
Dept. of Computing and Mathematical Sciences
Caltech, Pasadena, CA 91125, USA.
Email: {umang, katrina, schulman}@caltech.edu

Chaitanya Swamy
Dept. of Combinatorics & Optimization
Univ. Waterloo, Waterloo, ON N2L 3G1, Canada.
Email: cswamy@math.uwaterloo.ca

***Abstract*—The analysis of network routing games typically assumes, right at the onset, precise and detailed information about the latency functions. Such information may, however, be unavailable or difficult to obtain. Moreover, one is often primarily interested in enforcing a desirable target flow as the equilibrium by suitably influencing player behavior in the routing game. We ask whether one can achieve target flows as equilibria *without knowing the underlying latency functions*.**

Our main result gives a crisp positive answer to this question. We show that, under fairly general settings, one can efficiently compute *edge tolls* that induce a given target multicommodity flow in a nonatomic routing game using a *polynomial number of queries* to an *oracle* that takes candidate tolls as input and returns the resulting equilibrium flow. This result is obtained via a novel application of the ellipsoid method, and applies to arbitrary multicommodity settings and non-linear latency functions. Our algorithm extends easily to many other settings, such as (i) when certain edges cannot be tolled or there is an upper bound on the total toll paid by a user, and (ii) general nonatomic congestion games. We obtain tighter bounds on the query complexity for series-parallel networks, and single-commodity routing games with linear latency functions, and complement these with a query-complexity lower bound applicable even to single-commodity routing games on parallel-link graphs with linear latency functions. We also explore the use of *Stackelberg routing* to achieve target equilibria and obtain strong positive results for series-parallel graphs.

Our results build upon various new techniques that we develop pertaining to the computation of, and connections between, different notions of approximate equilibrium; properties of multicommodity flows and tolls in series-parallel graphs; and sensitivity of equilibrium flow with respect to tolls. Our results demonstrate that one can indeed circumvent the potentially-onerous task of modeling latency functions, and yet obtain meaningful results for the underlying routing game.

***Keywords*-Network routing; tolls; ellipsoid method; approximate equilibria; multicommodity flows; Stackelberg routing**

I. Introduction

Network routing games model settings where self-interested, uncoordinated users or agents route their traffic in a network—prominent examples include communication and transportation networks—and are extensively studied in Transportation Science and Computer Science (e.g., [1]–[3]). These games are typically described in terms of a directed graph $G = (V, E)$ modeling the network, a set of commodities specified by source-sink pairs and the volume of traffic routed between them modeling the users, and latency functions or delay functions $(l_e^*)_{e \in E}$ on the edges, with $l_e^*(x)$ modeling the delay on edge e when volume x of traffic is routed on it. The outcome of users' strategic behavior is described by an *equilibrium* traffic pattern, wherein no user may unilaterally deviate and reduce her total delay.

The analysis of network routing games typically takes the above specification as input, and thus assumes, right at the onset, precise, detailed information about the underlying latency functions. However, such precise information may be unavailable or hard to obtain, especially in large systems. In fact, the task of capturing observed delays via suitable delay functions is a topic of much research in itself in fields such as queuing theory and transportation science. Recognizing that the task of obtaining suitable latency functions is often really a means to facilitating a mathematical analysis of the underlying routing game, we ask whether one can sidestep this potentially-demanding task and analyze the routing game *without knowing the underlying latency functions*.

In routing games, there is often a central authority with limited ability to influence agent behavior by, e.g., imposing tolls on network edges. This ability can be used to alleviate the detrimental effects of selfish agent behavior, which might be expressed both in terms of the agents' costs (i.e., price of anarchy) and externalities not captured by these (e.g., pollution costs in a road network). Thus, a natural and well-studied goal in network routing games is to *induce a desirable target traffic pattern as an equilibrium* by suitably influencing agents' behavior. Such a target traffic pattern may be obtained by, e.g., limiting the traffic on every edge to a fraction of its capacity, or reducing the traffic near hospitals and schools. It is evident that obtaining the latency functions is only a means to the end goal of achieving the target traffic pattern. Our work sheds light on the question: *can one achieve this end without the means?*

A. Our contributions

We initiate a systematic study of network routing games from the perspective of achieving target equilibria without

knowing the latency functions. We introduce a *query model* for network routing games to study such questions, and obtain bounds on the query complexity of various tasks.

The query model: We are explicitly given the underlying network $G = (V, E)$, the set of commodities specified by the source-sink pairs and the demands to be routed between them, and the *target multicommodity flow* f^* that we seek to achieve. We *do not*, however, know the underlying latency functions $(l_e^*)_{e \in E}$. Instead, the only information that we can glean about the latency functions is via queries to a *black box* or *oracle* (e.g., simulation procedure) that outputs the equilibrium flow under a specified stimulus to the routing game. We investigate two methods for influencing agent behavior that have been considered extensively in the literature, which gives rise to two types of queries.

We primarily focus on the task of computing *edge tolls* to induce f^* (Section III). This yields the following query model: each query is a vector of tolls on the edges, and returns the equilibrium flow that results upon imposing these tolls. The goal is to minimize the number of queries required to compute tolls that yield f^* as the equilibrium.

We also explore, in Section IV, the use of *Stackelberg routing* to induce f^*. Here, we control an α fraction of the total traffic volume. Each query is a Stackelberg routing that specifies how this α-fraction is routed, and returns the equilibrium flow under this Stackelberg routing. The goal is to minimize the number of queries required to compute a Stackelberg routing that induces f^* as the equilibrium.

Our results and techniques: Our main result is a crisp and rather sweeping positive result showing that *one can always obtain tolls that induce a given target flow f^* with a polynomial number of queries* (Section III-A). With linear latency functions, our algorithm computes tolls that enforce f^* exactly (Theorem 4). With more general latency functions, such as convex polynomial functions, equilibria may be irrational, so it is not meaningful to assume that a query returns the exact equilibrium. Instead, we assume that each query returns a (suitably-defined) approximate equilibrium and obtain tolls that enforce a flow that is component-wise close to f^* (Theorem 8).

The chief technical novelty underlying these results is an unconventional application of the ellipsoid method. We view the problem as one where we are searching for the (parameters of the) true latency functions l^* and tolls that induce f^*. It is information-theoretically *impossible*, however, to identify l^* (or even get close to it) in the query model, since—as is the case even when G is a single edge—there may be no way of distinguishing two sets of latency functions. The key insight is that, notwithstanding this difficulty, if the current candidate tolls τ do not enforce f^*, then one can use the resulting equilibrium flow to identify a hyperplane that separates our current candidate (l, τ) from the true tuple (l^*, τ^*). This enables one to use the machinery of the ellipsoid method to obtain tolls enforcing f^* in a polynomial number of queries.

Our ellipsoid-method based algorithm is easily adapted to handle various generalizations (Section III-B). For instance, we can incorporate *any* linear constraints that tolls inducing f^* must satisfy, which one can separate over. This captures constraints where we disallow tolls on certain edges, or place an upper bound on the total toll paid by an agent. All our machinery extends seamlessly to the more-general setting of *nonatomic congestion games*. Finally, another notable extension is to the setting of *atomic routing games* under the assumption that the equilibrium is unique.

In Sections III-C and III-D, we devise algorithms with substantially improved query complexity for (a) multicommodity routing games on series-parallel (sepa) networks, and (b) single-commodity routing games on general networks, both with linear latency functions. For (a), we exploit the combinatorial structure of sepa graphs to design an algorithm with near-linear query complexity. We show that any toll-vector in a sepa graph can be converted into a simpler canonical form, which can be equivalently viewed in terms of certain labelings of the subgraphs of the sepa graph obtained via parallel joins; leveraging this yields an algorithm with near-linear query complexity. Our algorithm works more generally whenever we have an oracle that returns the (exact) equilibrium. For (b), we prove that (roughly speaking) the equilibrium flow is a linear function of tolls, and use linear algebra to infer the constants defining this linear map in $\tilde{O}(|E|^2)$ queries.

Complementing these upper bounds, we prove an $\Omega(|E|)$ *lower bound* (Theorem 25) on the query complexity of computing tolls that induce a target flow, even for single-commodity routing games on parallel-link graphs with linear delays. This almost matches the query complexity of our algorithm for sepa graphs.

En route to obtaining the above results, we prove various results that provide new insights into network routing games. For instance, we obtain results on: (a) the computation of approximate equilibria and their properties (Lemmas 6 and 7); (b) structural properties of tolls and multicommodity flows in sepa graphs (Section III-C); and (c) sensitivity of equilibrium flow with respect to tolls (Theorem 18). We believe that these results and the machinery we develop to obtain them are likely to find various applications.

In Section IV, we investigate the use of Stackelberg routing to induce a given target flow. Stackelberg routing turns out to be significantly harder to leverage than edge tolls in the query model. This is perhaps not surprising given that designing effective Stackelberg routing strategies is more difficult than computing suitable edge tolls, even in the standard setting where latency functions are given (e.g., [4], [5]). Nevertheless, we build upon the machinery that we develop for sepa graphs to give a rather efficient and general combinatorial algorithm that finds the desired Stackelberg routing using at most $|E|$ queries to an oracle

returning equilibrium flows. This applies to any strictly increasing latency functions, and in particular, to linear latency functions. (Observe that this is even better than our query-complexity bound for inducing flows via tolls on sepa graphs.) Moreover, our algorithm determines the Stackelberg routing of smallest volume that can induce f^*.

We obtain various lower bounds in Section V that allude to the difficulty of computing a Stackelberg routing in general networks that induces a target flow. One possible strategy for finding such a Stackelberg routing is to use the queries to infer an (approximately) "equivalent" set of delay functions l, in the sense that any Stackelberg routing yields the same (or almost the same) resulting equilibrium under the two sets of delay functions. Then, since given the latency functions, it is easy to compute a Stackelberg routing that induces a target flow (see Lemma 2), one can find the desired Stackelberg routing. Theorem 26 shows that such an approach cannot work: in the query model, any algorithm that learns even an approximately equivalent set of delay functions must make an *exponential* number of queries. Theorem 27 proves an orthogonal computational lower bound showing that determining the equivalence of two given sets of latency functions is an *NP*-hard problem. As in the case of tolls, along the way, we uncover a new result about the hardness of Stackelberg routing. We show that the problem of finding a Stackelberg routing that minimizes the equilibrium delay is *APX*-hard (Theorem 28). The query complexity of finding a Stackelberg routing in general networks that induces a target flow remains a very interesting open question.

Our results on tolls and Stackelberg routing demonstrate that one can indeed circumvent the potentially-onerous task of modeling latency functions, and yet obtain meaningful results for the underlying routing game. Our array of upper- and lower- bounds indicate the richness of the query model, and suggest a promising direction for further research.

B. Related work

Network routing games with nonatomic players—where each player controls an infinitesimal amount of traffic and there is a continuum of players—were first formally studied in the context of road traffic by Wardrop [6], and the equilibrium notion in such games is known as Wardrop equilibrium after him. Network routing games have since been widely studied in the fields of Transportation Science, Operations Research, and Computer Science; see, e.g., the monographs [2], [3] and the references therein. We limit ourselves to a survey of the results relevant to our work.

Equilibria are known to exist in network routing games, even with atomic players with splittable flow [3]. Nonatomic equilibria are known to be essentially unique, but this is not the case for atomic splittable routing games [7]. Equilibria in routing games are known to be inefficient, and considerable research has focused on quantifying this inefficiency in terms of the *price of anarchy* (PoA) [8] of the game, which measures, for a given objective, the worst-case ratio between the objective values of an equilibrium and the optimal solution. Tight bounds are known on the PoA for nonatomic routing games for the social welfare objective [9], [10].

Given the inefficiency of equilibria, researchers have investigated ways of influencing player behavior to alleviate this inefficiency. The most common techniques of influencing player behavior in routing games are imposing tolls on edges, and Stackelberg routing. Tolls are a classical means of congestion control, dating back to Pigou [11], and various results demonstrate their effectiveness for network routing games [1], [12]–[17]. Stackelberg routing is also well-studied, and it is known that this is much less effective in reducing the PoA. Whereas they can help in reducing the PoA to a constant for certain network topologies such as series-parallel graphs [16], this is not possible for general graphs [5]. Furthermore, it is *NP*-hard to compute the Stackelberg routing that minimizes the total cost at equilibrium, even for parallel-link graphs with linear delay functions [4]. All these results pertain to the setting where the latency functions are explicitly given.

To our knowledge, our query model has not been studied in the literature. It is useful to contrast our query model with work in *empirical game theory*, which also studies games when players' costs are not explicitly given. There, each query specifies a (pure or mixed) strategy-profile, and returns the (expected) cost of each player under this strategy profile. In contrast, in our query model, we observe the equilibrium flow instead of individual player delays. This is more natural in the setting of routing games: in the absence of knowledge of the latency functions, one may only be able to calculate player delays under a strategy profile by routing players along the stipulated paths (and then observing player delays); but this may be infeasible since one cannot in fact impose routes on self-interested players. Moreover, whereas our goal is to obtain a desirable outcome as the equilibrium, the focus in empirical game theory is to compute an (approximate) equilibrium. Generic approaches to generate strategy-profiles for this purpose, and examples where these have proved useful are discussed by Wellman [18]. Various papers study the complexity of computing an exact or approximate correlated equilibrium in multi-player games using both pure- and mixed-strategy queries [19]–[22]. Fearnley et al. [23] study the empirical-game-theory model for bimatrix games, congestion games, and graphical games, and obtain bounds on the number of queries required for equilibrium computation.

II. Preliminaries and notation

A *nonatomic routing game* (or simply a routing game) is denoted by a tuple $\Gamma = (G, l, \mathcal{K})$, where $G = (V, E)$ is a directed graph with m edges and n nodes, $l = (l_e)_{e \in E}$ is a vector of latency or delay functions on the edges, and $\mathcal{K} = \{(s_i, t_i, d_i)\}_{i \leq k}$ is a set of k triples denoting sources, sinks,

and demands for k *commodities*. The delay function $l_e : \mathbb{R}_+ \mapsto \mathbb{R}_+$ gives the delay on edge e as a function of the total flow on the edge. (Here, $\mathbb{R}_+$ is the set of nonnegative reals.) We assume that l_e is continuous, and strictly increasing. For each commodity i, the demand d_i specifies the volume of flow that is routed from s_i to t_i by self-interested agents, each of whom controls an infinitesimal amount of flow and selects an s_i-t_i path as her strategy. The strategies selected by the agents thus induce a multicommodity flow $(f^i)_{i\le k}$, where each $f^i = (f^i_e)_{e\in E}$ is an s_i-t_i flow of value d_i. That is, the vector $f^i = (f^i_e)_e$ satisfies:

$$\sum_{(v,w)\in E} f^i_{vw} - \sum_{(u,v)\in E} f^i_{uv} = 0 \quad \forall v \in V \setminus \{s_i, t_i\},$$
$$\sum_{(s,w)\in E} f^i_{sw} - \sum_{(u,s)\in E} f^i_{us} = d_i\,, \text{ and } f^i \ge 0\,.$$

We call $f = (f^i)_{i\le k}$ a feasible flow. We say that f is acyclic if $\{e : f^i_e > 0\}$ is acyclic for every commodity i. We overload notation and use f to also denote the total-flow vector $f = \sum_{i\le k} f^i$. For a path P, we use $f_P > 0$ to denote $f_e > 0$ for all $e \in P$. We sometimes refer to $\bigcup_i \{s_i, t_i\}$ as the terminals of the routing game or multicommodity flow. Given an s-t flow f, we use $|f|$ to denote the value of f.

Let $\mathcal{P}^i$ denote the collection of all s_i-t_i paths. Given a multicommodity flow $(f^i)_{i\le k}$ induced by the agents' strategies, the delay of an agent that selects an s_i-t_i path P is the total delay, $l_P(f) := \sum_{e\in P} l_e(f_e)$, incurred on the edges of P. Each agent in a routing game seeks to minimize her own delay. To analyze the resulting strategic behavior, we focus on the concept of a *Nash equilibrium*, which is a profile of agents' strategies where no individual agent can reduce her delay by changing her strategy, assuming other agents do not change their strategies. In routing games, this is formalized by the notion of *Wardrop equilibrium*.

Definition 1. A multicommodity flow $\hat{f}$ is a *Wardrop equilibrium* (or simply an equilibrium) of a routing game Γ if it is feasible and for every commodity i, and all paths $P,\ Q \in \mathcal{P}^i$ with $\hat{f}^i_P > 0$, we have $l_P(\hat{f}) \le l_Q(\hat{f})$. A Wardrop equilibrium can be computed by solving the following convex program:

$$\min\ \Phi(f) := \sum_e \int_0^{f_e} l_e(x)\,dx \quad \text{s.t.} \quad f = \sum_{i=1}^k f^i,$$
$$f^i \text{ is an } s_i\text{-}t_i \text{ flow of value } d_i \ \ \forall i = 1, \ldots, k. \quad (1)$$

Given a routing game Γ and a feasible flow f, define $D^i(l, f) := \min_{P\in\mathcal{P}^i} l_P(f)$ for each commodity i, and call an edge e a shortest-path edge for commodity i with respect to f if e lies on some path $P \in \mathcal{P}^i$ such that $l_P(f) = D^i(l, f)$. Let $\mathcal{S}^i(l, f)$ be the set of shortest-path edges for commodity i with respect to f.

Tolls, Stackelberg routing, and our query model: We investigate both the use of edge tolls and Stackelberg routing to induce a given target flow. Tolls are additional costs on the edges that are paid by every player that uses the edge. A vector of tolls $\tau = (\tau_e)_e \in \mathbb{R}^E_+$ on the network edges thus changes the delay function on each edge e to $l^\tau_e(x) := l_e(x) + \tau_e$, and so the delay of an agent who chooses P is now $l_P(f) + \tau(P)$, where $\tau(P) := \sum_{e\in P} \tau_e$. We use $f(l, \tau)$ to denote the equilibrium flow obtained with delay functions $l = (l_e)_e$ and tolls $\tau = (\tau_e)_e$. We say that τ enforces a multicommodity flow f with latency functions l if the total flow $f(l, \tau)_e = f_e$ on every edge e.

For Stackelberg routing, in keeping with much of the literature, we focus on single-commodity routing games. Given a single-commodity routing game $\Gamma = (G, l, (s, t, d))$ and a parameter $\alpha \in [0, 1]$, a central authority controls at most an α-fraction of the total s-t flow-volume d and routes this flow in any desired way, and then the remaining traffic routes itself selfishly. That is, a Stackelberg routing g is an s-t flow of value at most αd, which we call the Stackelberg demand. The Stackelberg routing g modifies the delay function on each edge e to $\tilde{l}_e(g; x) := l_e(x + g_e)$. The remaining $(1-\alpha)d$ volume of traffic routes itself according to a Wardrop equilibrium, denoted by $f(l, g)$, of the instance $(G, \tilde{l}, (1-\alpha)d)$. The total flow induced by a Stackelberg routing g is thus $g + f(l, g)$.

We shorten $f(l, \tau)$ to $f(\tau)$, and $f(l, g)$ to $f(g)$ when l is clear from the context.

In our query model, we are given the graph G, the commodity set $\mathcal{K} = \{(s_i, t_i, d_i)\}_{i\le k}$, and a feasible *target multicommodity flow* f^*. There is an underlying routing game $\Gamma = (G, l^*, \mathcal{K})$, to which we are given query access. If our method of influencing equilibria is via tolls, then the oracle takes a toll-vector τ as input and returns the equilibrium flow $f(l^*, \tau)$ or a (suitably-defined) approximate equilibrium. Our goal is to minimize the number of queries required to compute tolls τ^* such that $f(l^*, \tau^*) = f^*$.

If our method of influencing equilibria is via Stackelberg routing, then we are also given the parameter $\alpha \in [0, 1]$. Each query takes a Stackelberg routing g with $|g| \le \alpha d$ as input and returns the flow $f(l^*, g)$. Our goal is to minimize the number of queries required to compute a Stackelberg routing g^* of value at most αd such that $f(l^*, g^*) + g^* = f^*$, or determine that no such Stackelberg routing exists.

Properties of equilibria: The following facts about Wardrop equilibria, network tolls, and Stackelberg routing will be useful. Recall that the delay functions are nonnegative, continuous, and strictly increasing.

- A feasible flow f is an equilibrium flow iff $\sum_e (f_e - g_e) l_e(f_e) \le 0$ for every feasible flow g; see, e.g., [2]. Thus, the total-flow vector $(f_e)_e$ induced by an equilibrium flow is unique for strictly increasing delay functions.
- Every routing game admits an acyclic Wardrop equilib-

rium $\hat{f}$. If the delay functions are polytime computable, then one can solve (1) and compute: (i) $\hat{f}$ in polytime for linear delay functions; (ii) an acyclic flow f such that $\Phi(f) \leq \Phi(\hat{f}) + \epsilon$ in time $\mathrm{poly}\big(\text{input size}, \log(\frac{1}{\epsilon})\big)$. See, e.g., [2], for details.

- Every minimal feasible flow f is enforceable via tolls [13]–[15], where f is minimal if there is no feasible flow $g \neq f$ such that $g_e \leq f_e$ for every edge e. Given the edge delays $\big(l_e(f_e)\big)_e$, these tolls can be computed by solving an LP, and are rational provided the commodity demands $(d_i)_i$ and the delays $\big(l_e(f_e)\big)_e$ are rational.

Lemma 2 (essentially from [24])**.** *Let $(G, l, (d, s, t), \alpha)$ be a Stackelberg routing instance, and f^* be a feasible flow. Then, $f(g)+g = f^*$ for a Stackelberg routing g iff $g_e \leq f^*_e$ for every edge e, and $g_e = f^*_e$ for all $e \notin \mathcal{S}(l, f^*)$.*

Standard delay functions and encoding length: Our results hold for a broad class of underlying delay functions, that we now formally describe. Throughout, we use $\mathcal{I}$ denote the input size of the given routing game. We assume that we have an estimate U with $\log U = \mathrm{poly}(\mathcal{I})$ such that the target flow f^*, the parameters of the unknown true delay functions $(l^*_e)_e$, and the quantities that we seek to compute—tolls τ^* or the Stackelberg routing g^* inducing f^*—all have encoding length $O(\log U)$. So we may assume that every f^*_e, τ^*_e, g^*_e value is a multiple of $\frac{1}{U}$, and is at most U.

When considering non-linear delay functions, we assume that the l^*_es are convex polynomials of degree at most some known constant r. Given the $O(\log U)$ encoding length, we may assume that all coefficients lie in $[0, U]$ and and are multiples of $\frac{1}{U}$. We also assume that each $\frac{dl^*_e(x)}{dx} \geq \frac{1}{U}$ for all $x \geq 0$. We refer to such functions as *standard degree-r polynomials*. Under these mild conditions, it is easy to show that there is some constant $K := K(r) = \mathrm{poly}(U, \sum_i d_i)$ such that every delay function l^*_e satisfies

$$(x-y)\big(l^*_e(x) - l^*_e(y)\big) \leq \tfrac{\epsilon^2}{K} \implies |x-y| \leq \epsilon \quad \forall x, y, \epsilon \geq 0 \tag{2}$$

$$|l^*_e(x) - l^*_e(y)| \leq K|x-y| \quad \forall x, y \in [0, \textstyle\sum_i d_i] \tag{3}$$

$$l^*_e(2x) \leq K l^*_e(x) \quad \forall x \geq 0 \tag{4}$$

Properties (2)–(4) are referred to as *inverse-K-continuity*, *K-Lipschitz* and *K-growth boundedness*, respectively.

III. Inducing target flows via tolls

Recall that here we seek to compute tolls that enforce a given target flow f^* given black-box access to a routing game $\Gamma^* = (G, l^*, (s_i, t_i, d_i)_{i \leq k})$, i.e., without knowing l^*. Our main result is a crisp positive result showing that we can always achieve this end with a polynomial number of queries by leveraging the ellipsoid method in a novel fashion (Section III-A). Our algorithm computes tolls that enforce: (a) f^* exactly, for standard linear latency functions (where it is reasonable to assume that the black box returns the exact equilibrium); and (b) a flow that is component-wise close to f^*, for standard polynomial functions, where we now assume that each query only returns an approximate equilibrium (see Definition 5). We showcase the versatility of our algorithm by showing that it is easily adapted to handle various extensions (Section III-B).

In Sections III-C and III-D, we devise algorithms with significantly improved query complexity for multicommodity games on sepa networks, and single-commodity games on general networks. both with linear latencies.

A. An ellipsoid-based algorithm for general routing games

The ellipsoid method for finding a feasible point starts by containing the feasible region within a ball and generates a sequence of ellipsoids of successively smaller volumes. In each iteration, one examines the center of the current ellipsoid. If this is infeasible, then one uses a violated inequality to obtain a hyperplane, called a separating hyperplane, to separate the current ellipsoid center from the feasible region. One then generates a new ellipsoid by finding the minimum-volume ellipsoid containing the half of the current ellipsoid that includes the feasible region. We utilize the following well-known theorem about the ellipsoid method.

Theorem 3 ([25])**.** *Let $X \subseteq \mathbb{R}^n$ be a polytope described by constraints having encoding length at most M. Suppose that for each $y \in \mathbb{R}^n$, we can determine if $y \notin X$ and if so, return a hyperplane of encoding length at most M separating y from X. Then, we can use the ellipsoid method to find a point $x \in X$ or determine that $X = \emptyset$ in time $\mathrm{poly}(n, M)$.*

Linear latencies: We first consider the case where each $l^*_e(x)$ is a standard linear function $a^*_e x + b^*_e$, and our oracle returns the exact equilibrium flow induced by the input (rational) tolls. Thus, for every e, $a^*_e \in (0, U), b^*_e \in [0, U]$, and a^*_e, b^*_e are multiples of $\frac{1}{U}$. For a linear latency function $l(x) = ax + b$, we use l to also denote the tuple (a, b).

Theorem 4. *Given a target acyclic multicommodity flow f^* and query access to Γ^*, we can compute tolls that enforce f^* or determine that no such tolls exist, in polytime using a polynomial number of queries.*

Proof: We utilize the ellipsoid method and Theorem 3. In a somewhat atypical use of the ellipsoid method, we think of using it to search for the point $(a^*_e, b^*_e, \tau^*_e)_e$. As noted earlier, we *cannot*, however, hope to find (a^*, b^*, τ^*). But given the center $(\hat{l} = (\hat{a}_e, \hat{b}_e)_e, \hat{\tau})$ of the current ellipsoid, we show that if $\hat{\tau}$ does not induce f^*, then we can separate $(\hat{l}, \hat{\tau})$ from (a^*, b^*, τ^*). This implies that we terminate *either* with (a^*, b^*, τ^*) *or* with tolls $\hat{\tau}$ that induce f^*.

We first check if $\hat{a}, \hat{b}, \hat{\tau} \geq 0$, and if not, use the violated constraint as the separating hyperplane. Next, we use the black box to obtain $g = f(l^*, \hat{\tau})$. If $g = f^*$, then we are done. Otherwise, we obtain a separating hyperplane of

encoding length $\text{poly}(\mathcal{I})$ as follows. (Note that the encoding length of $(\hat{l}, \hat{\tau})$ is $\text{poly}(\mathcal{I})$.) We consider two cases.

Case 1: $f(\hat{l}, \hat{\tau}) \neq f^*$. Note that we can determine this without having to compute the equilibrium flow $f(\hat{l}, \hat{\tau})$. Since f^* is acyclic, we can efficiently find a commodity i, and s_i-t_i paths P, Q such that $f^*_P > 0$ and $\hat{l}_P(f^*) + \hat{\tau}(P) > \hat{l}_Q(f^*) + \hat{\tau}(Q)$. But since $f^* = f(l^*, \tau^*)$, we also have $l^*_P(f^*) + \tau^*(P) \leq l^*_Q(f^*) + \tau^*(Q)$. Thus, the inequality

$$l_P(f^*) + \tau(P) \leq l_Q(f^*) + \tau(Q)$$

where the parameters of l and τ are variables yields the desired separating hyperplane.

Case 2: $f(\hat{l}, \hat{\tau}) = f^*$. Since $g \neq f^*$ and is acyclic, we can again find efficiently i and paths $P, Q \in \mathcal{P}^i$ such that $g_P > 0$ and $\hat{l}_P(g) + \hat{\tau}(P) > \hat{l}_Q(g) + \hat{\tau}(Q)$. Since $g = f(l^*, \hat{\tau})$, we also have $l^*_P(g) + \hat{\tau}(P) \leq l^*_Q(g) + \hat{\tau}(Q)$. So the inequality $l_P(g) + \hat{\tau}(P) \leq l_Q(g) + \hat{\tau}(Q)$, where now *only* the l_es are variables, yields the separating hyperplane. ■

Polynomial latency functions and approximate equilibria: We now consider the setting where the latency functions $(l^*_e)_e$ are standard degree-r polynomials, where r is a known constant. We use l to also denote the tuple of coefficients of the polynomial given by l. Since the Wardrop equilibrium may now be irrational, it is unreasonable to assume that a query returns the equilibrium flow. So we assume that our black box returns an acyclic approximate equilibrium and show that we can nevertheless compute tolls that induce an equilibrium that is component-wise close to f^*. We first define approximate equilibria. Recall that $D^i(l, f) = \min_{P \in \mathcal{P}^i} l_P(f)$, and $l^\tau_e(x) := l_e(x) + \tau_e$ where $\tau = (\tau_e)_e$.

Definition 5. We say that a feasible flow f is an an *ϵ-equilibrium* of a routing game $(G, l, (s_i, t_i, d_i)_{i \leq k})$ if $\sum_e f_e l_e(f_e) \leq \sum_i d_i \big(D^i(l, f) + \epsilon\big)$.

Notice that our approximate-equilibrium notion is implied by the more-stringent (and oft-cited) condition requiring that if $f_P > 0$ for $P \in \mathcal{P}^i$ then $l_P(f) \leq D^i(l, f) + \epsilon$. Importantly, our notion turns out to be weak enough that one can argue that an acyclic ϵ-equilibrium can be computed in time $\text{poly}\big(\mathcal{I}, \log(\frac{1}{\epsilon})\big)$ for any $\epsilon > 0$, *which lends credence to our assumption that the black box returns an acyclic ϵ-equilibrium*, and yet is strong enough that one can leverage it within the framework of the ellipsoid method (see Theorem 8). Unless otherwise stated, when we refer to a routing game below, we assume that the latency functions satisfy (2)–(4), with $\log K$ being polynomially bounded.

Lemma 6. *Given a routing game with polytime-computable latency functions, one can compute an acyclic ϵ-equilibrium in time* $\text{poly}\big(\mathcal{I}, \log(\frac{1}{\epsilon})\big)$.

Lemma 7. *Let $\hat{f}$ be a Wardrop equilibrium and g be an ϵ-equilibrium of a routing game $(G, l, (s_i, t_i, d_i)_{i \leq k})$. Then, $\|g - \hat{f}\|_\infty := \max_e |g_e - \hat{f}_e| \leq \sqrt{K\epsilon \sum_i d_i}$.*

An *ϵ-oracle for tolls* is an oracle that given tolls τ as input, returns an ϵ-equilibrium of the routing game $(G, l^{*\tau}, (s_i, t_i, d_i)_{i \leq k})$ with encoding length $\text{poly}\big(\mathcal{I}, \log(\frac{1}{\epsilon})\big)$.

Theorem 8. *Let f^* be a target acyclic multicommodity flow f^* and $\delta > 0$. Let $\epsilon = \frac{\delta^2}{Kmk \sum_i d_i}$. Then, in time* $\text{poly}\big(\mathcal{I}, \log(\frac{1}{\delta})\big)$ *and using* $\text{poly}\big(\mathcal{I}, \log(\frac{1}{\delta})\big)$ *ϵ-oracle queries, we can compute tolls τ such that $\|f(l^*, \tau) - f^*\|_\infty \leq 2\delta$ or determine that no such tolls exist.*

Proof: We again use the ellipsoid method. Let $(\hat{l}, \hat{\tau})$ be the center of the current ellipsoid. Assume that $\hat{l}, \hat{\tau} \geq 0$ and each function $\hat{l}_e$ has slope at least $\frac{1}{U}$; otherwise, we can use a violated constraint as the separating hyperplane. We use the oracle with tolls $\hat{\tau}$ to obtain an acyclic ϵ-equilibrium g. Then, $\|g - f(l^*, \hat{\tau})\|_\infty \leq \sqrt{K\epsilon \sum_i d_i} = \frac{\delta}{\sqrt{mk}}$ by Lemma 7.

We can efficiently determine if $f(\hat{l}, \hat{\tau}) \neq f^*$, and if so, then as in Case 1 in the proof of Theorem 4, we can obtain a separating hyperplane of encoding length $\text{poly}\big(\mathcal{I}\big)$. So assume otherwise. Now we check if g is an $mk\epsilon$-equilibrium for the latency functions $(\hat{l}^{\hat{\tau}}_e)_e$. If so, then $\|g - f^*\|_\infty \leq \delta$ and so $\|f(l^*, \hat{\tau}) - f^*\|_\infty \leq 2\delta$ and we are done. Otherwise, we find a valid path-decomposition $x = (x_{i,P})_{i,P \in \mathcal{P}^i}$ of g having support of size at most mk. That is, we have $x \geq 0$, $\sum_{P \in \mathcal{P}^i} x_{i,P} = d_i$ for every commodity i, $\sum_i \sum_{P \in \mathcal{P}^i : e \in P} x_{i,P} = g_e$ for all e, and $\sum_i |\{P : x_{i,P} > 0\}| \leq mk$. We may assume that every non-zero $x_{i,P}$ value has encoding length that is polynomial in $\mathcal{I}$ and the size of g. Then $\sum_i \sum_{P \in \mathcal{P}^i} x_{i,P} \big(\hat{l}^{\hat{\tau}}_P(g) - D^i(\hat{l}^{\hat{\tau}}, g)\big) = \sum_e g_e \hat{l}^{\hat{\tau}}_e(g_e) - \sum_i d_i D^i(\hat{l}^{\hat{\tau}}, g) > mk\epsilon \sum_i d_i$ where the last inequality follows since g is not an $mk\epsilon$-equilibrium for $(\hat{l}^{\hat{\tau}}_e)_e$. Since the support of x has size at most mk, this implies that there is some commodity j and some path $R \in \mathcal{P}^j$ such that $x_{j,R} \big(\hat{l}^{\hat{\tau}}_R(g) - D^j(\hat{l}^{\hat{\tau}}, g)\big) > \epsilon \sum_i d_i$. Moreover, we can find such a j and path $R \in \mathcal{P}^j$ efficiently by simply enumerating the paths in the support of x. Let $Q \in \mathcal{P}^j$ be such that $\hat{l}^{\hat{\tau}}_Q(g) = D^j(\hat{l}^{\hat{\tau}}, g)$.

Since g is an ϵ-equilibrium for $(l^{*\hat{\tau}}_e)_e$, considering the path-decomposition of x, we have $\sum_i \sum_{P \in \mathcal{P}^i} x_{i,P} \big(l^{*\hat{\tau}}_P(g) - D^i(l^{*\hat{\tau}}, g)\big) \leq \epsilon \sum_i d_i$. Each term in this sum is nonnegative, so each term is at most $\epsilon \sum_i d_i$. In particular, we have $x_{j,R} \big(l^{*\hat{\tau}}_R(g) - l^{*\hat{\tau}}_Q(g)\big) \leq x_{j,R} \big(l^{*\hat{\tau}}_R(g) - D^j(l^{*\hat{\tau}}, g)\big) \leq \epsilon \sum_i d_i$. So the inequality $x_{j,R} \big(l_R(g) + \hat{\tau}(R) - l_Q(g) - \hat{\tau}(Q)\big) \leq \epsilon \sum_i d_i$, with l_es as the variables, is valid for (l^*, τ^*) but is violated by $(\hat{l}, \hat{\tau})$. This yields a separating hyperplane of encoding length $\text{poly}\big(\mathcal{I}, \log(\frac{1}{\epsilon})\big)$. ■

B. Extensions

Our ellipsoid-based algorithm easily extends to various more-general settings including the following.

- *Linear constraints on tolls.* Here, we require that the tolls τ^* imposing the target flow f^* should lie in some polyhedron X, where X is specified via a separation

oracle. This is rich enough to model: (i) a subset F of edges cannot be tolled; and (ii) the total toll paid by any player under the flow f^* is at most a given budget B.

- *General nonatomic congestion games.* This is a generalization of network routing games, where the graph is replaced by an arbitrary set E of resources, and $\mathcal{P}^i \subseteq 2^E$ is the strategy-set associated with player-type i.
- *Atomic splittable routing games.* Here, each commodity i represents a *single* player who controls d_i volume of flow and her strategy is to choose an s_i-t_i flow f^i of value d_i. The cost incurred by a player i under a strategy profile $f = (f^i)_{i \le k}$ is $\sum_e f^i_e l_e(f_e)$. Our results extend here if we assume that for all valid choices of parameters of the latency functions and tolls (as encountered during the ellipsoid method), the underlying atomic splittable routing game has a unique Nash equilibrium.

C. An improved algorithm for series-parallel networks

We give an algorithm for series-parallel networks with $\tilde{O}(m)$ query complexity. This significantly improves upon the ellipsoid-based algorithm, and almost matches the linear lower bound proved in Theorem 25.

Theorem 9. *On two-terminal series-parallel graphs, one can compute in polytime tolls that induce a given target multicommodity flow f^* using $\tilde{O}(m)$ queries to an oracle that returns the equilibrium flow. Thus, we obtain $\tilde{O}(m)$ query complexity for multicommodity routing games with standard linear delay functions.*

We first recall some relevant details about series-parallel graphs. A *two-terminal directed series-parallel graph*, abbreviated series-parallel (sepa) graph, with terminals s and t is defined inductively as follows. A basic sepa graph is a directed edge (s, t). Given two sepa graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with terminals s_1, t_1 and s_2, t_2 respectively, one can create a new sepa graph $G = (V, E)$ as follows. A *series join* of G_1 and G_2 yields the graph obtained by identifying t_1 and s_2, with terminals $s = s_1$ and $t = t_2$. A *parallel join* of G_1 and G_2 yields the graph obtained by identifying s_1 and s_2, and t_1 and t_2; its terminals are $s = s_1 = s_2$ and $t = t_1 = t_2$.

For every series-parallel graph $G = (V, E)$, the recursive construction naturally yields a binary *decomposition tree*, whose leafs are edges of G, and each internal node specifies a series- or a parallel- join. Each node of the tree also represents a subgraph of the G (obtained by performing the joins specified by the subtree rooted at that node), which is also clearly a sepa graph. In the sequel, we fix a decomposition tree corresponding to G, and by "subgraph of G" we mean a subgraph corresponding to a node of this decomposition tree. Given a subgraph H, we use s_H, t_H to denote its two terminals, which we sometimes call source and sink of H respectively, and $\mathcal{P}(H)$ to denote the set of all s_H-t_H paths. Let $\mathcal{H}$ be the collection of subgraphs corresponding to parallel-join nodes of the decomposition tree. (Note that $|\mathcal{H}| \le m$.) For each $H \in \mathcal{H}$ obtained via the parallel join of H_1 and H_2, we identify one of these as the "left" subgraph H_L and the other as the "right" subgraph H_R. Let $\mathcal{P}$ denote the set of all s-t paths, where $s = s_G,\ t = t_G$.

Proof outline: We first give some intuition for the proof of Theorem 9. It is useful to consider the simplest case of a graph with two parallel edges. Observe that any target flow can be obtained by varying the *difference* in tolls on these two edges. Further, the correct difference in tolls can be obtained by a binary search. Our key insight is that this intuition can be extended to series-parallel graphs via a suitable transformation of tolls. We show that tolls required to obtain a target flow can actually be described by the difference in tolls for each pair of parallel subgraphs, and then use binary search to obtain the correct differences that yield the target flow.

Formally, we show that any edge tolls in a sepa graph can in fact be transformed into certain canonical tolls that are defined in terms of subgraphs (Claim 11). Further, formalizing the intuition that what is relevant is only the difference in tolls on parallel subgraphs, we show that canonical tolls are in fact equivalent to labels on subgraphs $H \in \mathcal{H}$ (Lemma 12), where the label on subgraph $H \in \mathcal{H}$ stores the difference in the canonical tolls of subgraphs H_L and H_R whose parallel-join yields H.

Thus, our problem reduces to finding the correct labels on subgraphs $H \in \mathcal{H}$, which we do via binary search. For this, we establish certain structural properties of multicommodity flows in sepa graphs (Lemma 14). We leverage these to argue that if the canonical edge-tolls obtained from our current labels do not enforce the target flow, then we can find a subgraph $H \in \mathcal{H}$ and deduce whether its label should be increased or decreased. The query complexity is thus at most $|\mathcal{H}|$ times a logarithmic term depending on the accuracy required and the parameters of the routing game.

The presence of multiple commodities complicates things, since in the particular decomposition tree that we fix for G, all edges in a subgraph may be shortest-path edges for one commodity but not for another. Thus creates problems with the binary search since Claim 15 may not hold. We handle this by first arguing that there always exist tolls enforcing f^* such that *every* s-t path, and hence every s_i-t_i path is a shortest-path under edge costs $(l^{*\tau^*}_e(f^*_e))_e$ (Claim 10). All omitted proofs appear in the full version.

We believe that our structural insights into tolls and multicommodity flows on sepa graphs are of independent interest. In fact, our results on sepa graphs play an important role in our algorithm for inducing target flows via Stackelberg routing in Section IV.

Claim 10. *For $\Gamma^* = (G, l^*, (s_i, t_i, d_i)_{i<k})$ and target flow f^* there exist tolls $\tau^* \in \mathbb{R}^E_+$ such that: (i) $\min_{P \in \mathcal{P}} \tau^*(P) = 0$; (ii) $l^*_P(f^*) + \tau^*(P) = l^*_Q(f^*) +$*

$\tau^*(Q)$ *for every* i *and paths* $P, Q \in \mathcal{P}^i$*; (iii)* $f(l^*, \tau^*) = f^*$.

Claim 11. *For any tolls* $\tau \in \mathbb{R}^E_+$ *on the edges of* G*, there exist* $\alpha \in \mathbb{R}^E_+$ *such that: (i)* $\tau(P) = \alpha(P)$ *for all* $P \in \mathcal{P}$*, and (ii) for every subgraph* H *and every edge* $e = (s_H, v) \in E(H)$*,* $\alpha_e \geq \min_{P \in \mathcal{P}(H)} \alpha(P)$.

We call tolls $\alpha \in \mathbb{R}^E_+$ that satisfy property (ii) of Claim 11 *canonical tolls*. Thus, any edge tolls can be modified to obtain canonical edge tolls α. These in turn can be mapped to a *labeling* (L, Δ), where $\Delta = (\Delta_H)_{H \in \mathcal{H}} \in \mathbb{R}^{\mathcal{H}}_+$, by setting $L = \min_{P \in \mathcal{P}} \alpha(P)$, and $\Delta_H = \min_{P \in \mathcal{P}(H_L)} \alpha(P) - \min_{P \in \mathcal{P}(H_R)} \alpha(P)$ for all $H \in \mathcal{H}$. Lemma 12 shows that this mapping is in fact invertible. Given the labeling (L, Δ) we can obtain canonical edge tolls α as follows.

M1. Initialize $\alpha_e = 0$ for all e.
M2. We consider subgraphs in $\mathcal{H}$ starting from the leaves and moving up to the root. When considering $H \in \mathcal{H}$, we set $\alpha_e = \alpha_e + \max\{0, \Delta_H\}$ for all $e = (s_H, v) \in E(H_L)$, and $\alpha_e = \alpha_e + \max\{0, -\Delta_H\}$ for all $e = (s_H, v) \in E(H_R)$.
M3. Finally, we set $\alpha_e = \alpha_e + L$ for all $e = (s, v) \in E$.

Lemma 12. *Let* (L, Δ) *be the labeling obtained from some canonical tolls* $\alpha \in \mathbb{R}^E_+$*, and* β *be the tolls obtained from* (L, Δ) *by the above procedure. Then* $\alpha = \beta$.

Definition 13. Given multicommodity flows f and $\tilde{f}$, we call a pair H_1, H_2 of subgraphs, $(f, \tilde{f})$*-discriminating* if: (i) the parallel-join of H_1 and H_2 is a subgraph in $\mathcal{H}$; and (ii) $f_e > \tilde{f}_e$ for all $e \in E(H_1)$, and $f_e \leq \tilde{f}_e$ for all $e \in E(H_2)$.

Lemma 14. *Let* f *and* $\tilde{f}$ *be two feasible multicommodity flows for* $(G, (s_i, t_i, d_i)_{i \leq k})$*. If* $f \neq \tilde{f}$*, then there exists an* $(f, \tilde{f})$*-discriminating pair of subgraphs.*

Claim 15. *Let* $\hat{f} = f(l^*, \tau)$*. If there is a subgraph* H *with* $\hat{f}_e > f^*_e \ \forall e \in E(H)$ *then for some* i*, every* s_H*-*t_H *path is part of a shortest* s_i*-*t_i *path under edge costs* $(l^{*\tau}_e(\hat{f}_e))_e$.

We now describe the algorithm leading to Theorem 9. Let τ^* be tolls given by part (b) of Claim 10 and $(0, \Delta^*)$ be the labeling obtained from τ^*. We may assume that $\tau^*_e \in [0, U']$ and is a multiple of $\frac{1}{U'}$ for all e, where $U' = m \operatorname{poly}(U, \sum_i d_i)$. E.g., with standard linear latencies, since every $f^*_e, a^*_e, b^*_e \in [0, U]$ and is a multiple of $\frac{1}{U}$, we can take $U' = \max\{U^2, mK \sum_i d_i\}$.

T1. Initialize, $L_H = -mU'$, $U_H = mU'$, $\Delta_H = 0$ for all $H \in \mathcal{H}$. Let $L = 0$. Let $M = m \log(8mU'^2)$.
T2. For $r = 1, \ldots, M$, we do the following. Map (L, Δ) to canonical tolls α as described in steps M1–M3. Query the oracle to obtain $\hat{f} = f(l^*, \alpha)$. If $\hat{f} = f^*$, then exit the loop. Otherwise, find an $(\hat{f}, f^*)$-discriminating pair of subgraphs H_1, H_2 (which exists by Lemma 14). Let H be the parallel join of H_1, H_2. If $H_1 = H_L$, update $L_H \leftarrow \Delta_H$, else update $U_H \leftarrow \Delta_H$. If $|U_H - L_H| < \frac{1}{U'}$, set Δ_H to be the multiple of $\frac{1}{U'}$ in $[L_H, U_H]$; else update $\Delta_H = (L_H + U_H)/2$.
T3. Return tolls α.

Proof Sketch of Theorem 9: We show that the binary search performed by the algorithm is valid; that is, we maintain the invariant that $\Delta^*_H \in [L_H, U_H]$ for all $H \in \mathcal{H}$. Under the assumptions on τ^*, one can show that $\Delta^*_H \in [-mU', mU']$ and is a multiple of $\frac{1}{U'}$, for all $H \in \mathcal{H}$, so this holds at the beginning. For the H_1, H_2 pair chosen in an iteration of step T2, we utilize Claim 15 to show that we must have $\alpha_{H_1} - \alpha_{H_2} < \alpha^*_{H_1} - \alpha^*_{H_2}$. Hence, our update for H at the end of the iteration preserves the invariant. Thus, since $|\mathcal{H}| \leq m$, the binary search takes at most M iterations. ■

Observe that the above algorithm in fact works whenever we have a "sign oracle" that given input tolls τ and a flow f^*, returns the sign of $f(l^*, \tau)_e - f^*_e$ for all edges e.

D. Single-commodity, linear-delay routing games

We devise an algorithm with nearly quadratic query complexity here. We show that flows are linear functions of tolls and infer this linear map using $\tilde{O}(m^2)$ queries.

Theorem 16. *For a single-commodity routing game* Γ *with standard linear delay functions, tolls that enforce* f^* *can be obtained in at most* $\tilde{O}(m^2)$ *queries.*

We assume that $f^* > 0$; otherwise, we can impose very large tolls on any edge with $f^*_e = 0$, effectively deleting such edges. Let $l_e(x) = a_e x + b_e$. Define $l_{\max}(x) := \max_{e \in E}(a_e x + b_e)$, and $\kappa(x) = x^2/Kd$ (where d is the s-t flow volume). Define the *support* of a flow f to be the set of edges with strictly positive flow. We will use negative tolls in our proof; however, by Claim 17, this is just a notational convenience. Similar arguments were used in [26] to show boundedness of tolls. Note that f^* is acyclic.

Claim 17. *For a single-commodity routing game and tolls* τ*, there exist tolls* $\tau' \geq 0$ *so that* $f(\tau) = f(\tau')$ *and* $\tau'_{e'} \leq \tau_{e'} + \sum_{e: \tau_e < 0} |\tau_e|$ *for all* e'*. If the graph is acyclic,* τ' *can be obtained without knowledge of the delay functions.*

Proof outline: We show that if the support of the equilibrium flow remains fixed, the equilibrium flow is a linear function of the tolls. Thus if we can obtain tolls τ so that the support of $f(\tau)$ is the same as f^*, we can solve a linear system of equations to obtain tolls that enforce f^*. Our algorithm consists of the following two steps.

Step 1: Enforcing the correct support. We first obtain tolls τ so that $f_e(\tau) > 0 \Leftrightarrow f^*_e > 0$. Since $f^* > 0$ by assumption, one direction of the implication is already satisfied. The other direction is roughly by binary search, described in Lemma 19: we pick an edge r that does not yet have flow, and decrease the toll on r until it has positive flow at equilibrium. However, increasing the flow on edge r may decrease flow on the other edges. To maintain monotonicity of the support of the equilibrium flow, we use results about the sensitivity of equilibrium flow. In fact, this step has quadratic query complexity, while the second step is linear.

Step 2: Obtaining the target flow f^*. We use Lemma 21 which establishes the linearity of equilibrium flow as a

function of tolls, if the support of the equilibrium flow does not change. We obtain the coefficients of this linear map by querying the oracle with a small toll on each edge. The query complexity of this step is thus linear.

We begin by showing the monotonicity and sensitivity of equilibrium flow as a function of tolls. Using these properties we argue that we can iteratively obtain positive flow on every edge (Lemma 20), and thus implement Step 1. We note that while some results in [27], [28] are similar to Theorem 18 and Lemma 21, they need to be modified suitably for our results. Let $\mathbb{1}_e \in \mathbb{R}^E$ be the vector with value 1 in coordinate e, and 0 elsewhere. Throughout, Γ is a single-commodity routing game with standard linear latencies.

Theorem 18. *For routing game Γ and any edge r,*
(i) $\|f(0) - f(\mathbb{1}_r\kappa(\epsilon))\|_\infty \leq \epsilon$, *(ii)* $f(\tau)$ *is continuous,*
(iii) $f_r(\mathbb{1}_r\delta) \leq f_r(0)$ *for all* $\delta > 0$*, and*
(iv) $|f_r(-\mathbb{1}_r\delta) - f_r(0)| \geq \|f(-\mathbb{1}_r\delta) - f(0)\|_\infty$ *for all* $\delta > 0$.

Lemma 19. *Let Γ be a routing game. Fix $0 < \delta \leq d$. For tolls τ, let $S := \{e : f_e(\tau) \geq \delta\}$. Let r be an edge not in S. Using $\log(-N/\kappa(\delta/3))$ queries, where $N := \min_{P\in\mathcal{P}} \tau(P) - \min_{P\in\mathcal{P}: r\in P} \tau(P) - ml_{\max}(d) < 0$, we can compute $\alpha \in [N, 0]$ so that for $\tau' = \tau + \alpha\mathbb{1}_r$, we have $f_e(\tau') \geq \delta/3$ for all $e \in S \cup \{r\}$.*

Lemma 20. *Using $O(m^2 \log(3mKl_{\max}(d))$ queries, we can compute tolls τ so that $f_e(\tau) \geq d/3^m$ for all e.*

Proof Sketch: We repeatedly use Lemma 19. In iteration i, with current tolls τ, if $S_i = \{e : f_e(\tau) \geq d/3^i\} \neq E$, then we use Lemma 19 to ensure that $S_{i+1} \supsetneq S_i$. We argue that the N needed to do this is at least $-m2^{i-1}l_{\max}(d)$. This yields the stated query complexity. ■

Lemma 21. *Let Γ be a routing game, tolls $\tau^{(1)}$ be such that $f(\tau^{(1)}) > 0$. There exists a matrix $\beta = (\beta_{e,e'})_{e,e'\in E}$ so that for any tolls τ,*
(i) $f(\tau + \tau^{(1)}) > 0 \implies f(\tau + \tau^{(1)}) = f(\tau^{(1)}) + \beta\tau$*;*
(ii) $f(\tau^{(1)}) + \beta\tau > 0 \implies f(\tau + \tau^{(1)}) = f(\tau^{(1)}) + \beta\tau$.

Proof of Theorem 16: We first use Lemma 20 to obtain tolls $\tau^{(1)}$ such that $f_e(\tau^{(1)}) \geq d/3^m$ for all e. Next, we use Lemma 21 (i) to obtain the matrix β. We use m queries, each applying an additional toll τ (relative to $\tau^{(1)}$) of $\kappa(d/3^{m+1})$ on a distinct edge (which ensures that $f(\tau^{(1)} + \tau) > 0$ by Theorem 18 (i)). Finally, we solve the system $f^* = f(\tau^{(1)}) + \beta\tau^{(2)}$ for $\tau^{(2)}$, and return the tolls $\tau^{(1)} + \tau^{(2)}$. Correctness follows immediately from Lemma 21: the system solved is feasible since $f^* = f(\tau^*) > 0$, and so $f^* = f(\tau^{(1)}) + \beta(\tau^* - \tau^{(1)})$; since $f(\tau^{(1)}) + \beta\tau^{(2)} = f^* > 0$, we have $f(\tau^{(1)} + \tau^{(2)}) = f^*$. ■

IV. Inducing target flows via Stackelberg routing on series-parallel graphs

We are now given a single-commodity routing game $\Gamma^* = (G, l^*, (s, t, d))$ on a sepa graph G, $\alpha \in [0, 1]$ and a target flow f^*, and we seek an s-t flow g of value of at most αd so that $g + f(l^*, g) = f^*$, if one exists. We abbreviate $f(l^*, g)$ to $f(g)$. We devise an algorithm that computes a Stackelberg routing inducing f^* using at most m queries. The flow g we compute is in fact of minimum value among all Stackelberg flows that induce f^*. (So either g is the desired Stackelberg flow, or none exists if $|g| > \alpha d$.) Our algorithm works for arbitrary increasing delay functions provided, as in Section III-C, we have an oracle that returns the correct sign of $((f(g) + g)_e - f^*)_e$ given a Stackelberg routing g. In particular, the algorithm works for increasing linear latency functions.

As before, we fix a decomposition tree for G, and a subgraph refers to a subgraph corresponding to a node of this tree. For a flow f and subgraph H, let f_H denote $(f_e)_{e\in E(H)}$. Similar to Definition 13, we define the notion of a good pair of subgraphs.

Definition 22. Given s-t flows f, $\tilde{f}$, we call a pair of subgraphs H_1, H_2 $(f, \tilde{f})$-*good* if: (i) the parallel-join of H_1, H_2 is a subgraph; (ii) $f_e \geq \tilde{f}_e$ $\forall e \in E(H_1)$, $f_e \leq \tilde{f}_e$ $\forall e \in E(H_2)$; and (iii) $|f_{H_1}| > |\tilde{f}_{H_1}|$, $|f_{H_2}| < |\tilde{f}_{H_2}|$.

Lemma 23. *Let g be any Stackelberg routing. If $f(g) + g \neq f^*$, there exists an $(f(g) + g, f^*)$-good pair of subgraphs.*

Theorem 24. *We can compute a Stackelberg flow of minimum value that induces f^* in at most m queries.*

Proof Sketch: The algorithm is quite simple. We keep track of the set $\bar{S}$, initialized to $\emptyset$, of edges not on any shortest s-t path under edge costs $(l^*_e(f^*_e))_e$. By Lemma 2, $\bar{S}$ must be saturated by any Stackelberg routing that induces f^*. So we find the flow $g \leq f^*$ of minimum value that saturates $\bar{S}$. If $f(g) + g \neq f^*$, we find an $(f(g) + g, f^*)$-good pair of subgraphs H_1, H_2. We argue that H_2 cannot contain any shortest s-t path edges, so we update $\bar{S}$ and repeat. ■

V. Lower bounds

We obtain various lower bounds on the complexity of problems motivated by our query model. We begin with a lower bound on the query-complexity of computing tolls.

Theorem 25. *Any deterministic algorithm that computes tolls to enforce f^* requires $\Omega(m)$ queries, even for a single commodity on parallel links with linear delay functions.*

A natural question that arises from our query model is whether one can infer the latency functions. While this is impossible even for a single edge, perhaps one can infer an "equivalent" set of latency functions. We show lower bounds on the query- and computational- complexity of this problem

for Stackelberg routing. As a by-product, we obtain that the problem of finding a Stackelberg flow g that minimizes the equilibrium delay is *APX*-hard. We formalize equivalence as follows: given a graph G, demand d to be routed from s to t, and $\alpha \in [0,1]$, two sets of latency functions $\{l_e^1\}_e$, $\{l_e^2\}_e$ are ϵ-*equivalent* if $\|f(l^1,g)-f(l^2,g)\|_\infty \leq \epsilon$ for every Stackelberg routing g with $|g| \leq \alpha d$.

Theorem 26. *Any deterministic algorithm that determines ϵ-equivalence for $\epsilon \leq 1/16$ has exponential query complexity.*

Theorem 27. *It is NP-complete to decide if two (explicitly given) delay functions are ϵ-equivalent, for any $\epsilon < \frac{1}{2}$.*

Theorem 28. *The problem of minimizing $D(f(g))$ over all Stackelberg flows g is $\left(\frac{4}{3}-\epsilon\right)$-inapproximable, for any $\epsilon > 0$.*

ACKNOWLEDGMENTS

We thank Éva Tardos for useful discussions. UB was supported by a Linde/SISL postdoctoral fellowship and NSF grants CNS-0846025, CCF-1101470 and EPAS-1307794. KL was supported by the Charles Lee Powell Foundation and a Microsoft Faculty Fellowship. LJS was supported by NSF grants 1038578 and 1319745, and was partly at the Simons Institute at UC Berkeley. CS was supported by NSERC grant 32760-06, an NSERC Discovery Accelerator Supplement Award, and an Ontario Early Researcher Award.

REFERENCES

[1] M. Beckman, C. McGuire, and C. B. Winsten, "Studies in the economics of transportation," Tech. Rep., 1956.

[2] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.

[3] ——, "Routing games," in *Algorithmic Game Theory*, N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, Eds. Cambridge University Press, 2007.

[4] ——, "Stackelberg scheduling strategies," *SIAM J. Comput.*, vol. 33, no. 2, pp. 332–350, 2004.

[5] V. Bonifaci, T. Harks, and G. Schäfer, "Stackelberg routing in arbitrary networks," *MOR*, vol. 35, pp. 330–346, 2010.

[6] J. G. Wardrop, "Some theoretical aspects of road traffic research," in *ICE Proceedings: Engineering Divisions*, vol. 1, no. 3. Thomas Telford, 1952, pp. 325–362.

[7] U. Bhaskar, L. Fleischer, D. Hoy, and C. Huang, "Equilibria of atomic flow games are not unique," in *SODA*, 2009, pp. 748–757.

[8] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *Proceedings of 16th STACS*, 1999, pp. 404–413.

[9] T. Roughgarden, "The price of anarchy is independent of the network topology," *JCSS*, vol. 67, pp. 341–364, 2003.

[10] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *J. ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[11] A. C. Pigou, *The Economics of Welfare*. Macmillan, 1920.

[12] R. Cole, Y. Dodis, and T. Roughgarden, "Pricing network edges for heterogeneous selfish users," in *STOC*, 2003, pp. 521–530.

[13] L. Fleischer, K. Jain, and M. Mahdian, "Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games," in *FOCS*, 2004, pp. 277–285.

[14] G. Karakostas and S. G. Kolliopoulos, "Edge pricing of multicommodity networks for heterogeneous users," in *FOCS*, 2004, pp. 268–276.

[15] H. Yang and H.-J. Huang, "The multi-class, multi-criteria traffic network equilibrium and systems optimum problem," *Trans. Res. B: Methodological*, vol. 38, pp. 1–15, 2004.

[16] C. Swamy, "The effectiveness of Stackelberg strategies and tolls for network congestion games," *ACM Transactions on Algorithms*, vol. 8, no. 4, p. 36, 2012.

[17] H. Yang and X. Zhang, "Existence of anonymous link tolls for system optimum on networks with mixed equilibrium behaviors," *Trans. Res. B*, vol. 42, pp. 99–112, 2008.

[18] M. P. Wellman, "Methods for empirical game-theoretic analysis," in *Proc., National Conf. on AI*, vol. 21, 2006, p. 1552.

[19] C. Papadimitriou and T. Roughgarden, "Computing correlated equilibria in multi-player games," *JACM*, vol. 55, p. 14, 2008.

[20] A. Jiang and K. Leyton-Brown, "Polynomial-time computation of exact correlated equilibria in compact games," *Games and Economic Behavior*, 2013, to appear.

[21] Y. Babichenko, S. Barman, and R. Peretz, "Simple approximate equilibria in large games," in *EC*, 2014, pp. 753–770.

[22] S. Hart and N. Nisan, "The query complexity of correlated equilibria," 2013, CS arXiv.

[23] J. Fearnley, M. Gairing, P. W. Goldberg, and R. Savani, "Learning equilibria of games via payoff queries," in *ACM Conference on Electronic Commerce*, 2013, pp. 397–414.

[24] A. Kaporis and P. Spirakis, "The price of optimum in Stackelberg games on arbitrary single commodity networks and latency functions," *TCS*, vol. 410, pp. 745–755, 2009.

[25] M. Grötschel, L. Lovász, and L. Schrijver, "Geometric algorithms and combinatorial optimization," *Algorithms and Combinatorics*, vol. 2, pp. 1–362, 1993.

[26] L. K. Fleischer, "Linear tolls suffice: New bounds and algorithms for tolls in single source networks," *Theor. Comput. Sci.*, vol. 348, no. 2-3, pp. 217–225, 2005.

[27] S. Dafermos and A. Nagurney, "Sensitivity analysis for the asymmetric network equilibrium problem," *Mathematical programming*, vol. 28, no. 2, pp. 174–184, 1984.

[28] R. Cole, Y. Dodis, and T. Roughgarden, "How much can taxes help selfish routing?" *JCSS*, pp. 444–467, 2006.