Equilibrium Computation in Atomic Splittable Routing Games

Umang Bhaskar[1]
Tata Institute of Fundamental Research, Mumbai, India
umang@tifr.res.in

Phani Raj Lolakapuri
Tata Institute of Fundamental Research, Mumbai, India
phaniraj@tcs.tifr.res.in
https://orcid.org/0000-0003-1593-9654

Abstract

We present polynomial-time algorithms as well as hardness results for equilibrium computation in atomic splittable routing games, for the case of general convex cost functions. These games model traffic in freight transportation, market oligopolies, data networks, and various other applications. An atomic splittable routing game is played on a network where the edges have traffic-dependent cost functions, and player strategies correspond to flows in the network. A player can thus split its traffic arbitrarily among different paths. While many properties of equilibria in these games have been studied, efficient algorithms for equilibrium computation are known for only two cases: if cost functions are affine, or if players are symmetric. Neither of these conditions is met in most practical applications. We present two algorithms for routing games with general convex cost functions on parallel links. The first algorithm is exponential in the number of players, while the second is exponential in the number of edges; thus if either of these is small, we get a polynomial-time algorithm. These are the first algorithms for these games with convex cost functions. Lastly, we show that in general networks, given input C, it is NP-hard to decide if there exists an equilibrium where every player has cost at most C.

2012 ACM Subject Classification Theory of computation → Network games

Keywords and phrases Routing Games, Equilibrium Computation, Convex costs, Splittable flows

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.58

Related Version A full version of the paper is available at [6], `https://arxiv.org/abs/1804.10044`.

1 Introduction

The problem of equilibrium computation, particularly efficient computation, is the cornerstone of algorithmic game theory, and is an area where researchers have had many successes. In many games, we have a good understanding of where the boundaries of computation lie, including normal-form games [9], markets [11], and congestion games [10]. The study of equilibrium computation has had a significant impact on algorithms, contributing new techniques and complexity classes.

[1] This work was partly funded by a Ramanujan Fellowship.

In this paper, we are interested in equilibrium computation in atomic splittable routing games (ASRGs) with convex cost functions. These games are used to model many applications, including freight transportation, market oligopolies, and data networks (e.g., [8, 21]). In an ASRG, we are given a network with cost functions on the edges, and k players. Each player i has a source s_i, destination t_i, and a fixed demand v_i. Each player needs to transport its demand from its source to its destination at minimum cost, and is free to split the demand along multiple paths. Each player thus computes a minimum cost s_i-t_i flow, given the strategy of the other players.

The fact that each player can split its flow along multiple paths is what differentiates these from weighted congestion games. This freedom reduces the combinatorial structure of the game, making ASRGs harder to analyze. For example, equilibria in ASRGs may be irrational. While equilibrium computation in (unsplittable) congestion games is well-studied, much less is known about ASRGs. In fact, properties of ASRGs apart from equilibrium computation have been studied. We know tight bounds on the price of anarchy [8, 14, 25], and can characterize games with multiple equilibria [4].

However for equilibrium computation little is known. We know of only two cases when an equilibrium can be efficiently computed – when cost functions are affine, or when players are symmetric, i.e., they have the same source, destination, and demand [8, 16]. These conditions are hardly ever met in practice. We know of no hardness results for this problem. A number of iterative algorithms for equilibrium computation are proposed, and sufficient conditions for convergence are given by Marcotte [20]. Further, it is implicit in a paper by Swamy that one can compute equilibrium efficiently, given the total flow on each edge [27].

Computing equilibria in ASRGs is an interesting theoretical challenge as well. In some regards, properties of pure Nash equilibria in ASRGs resemble mixed Nash equilibria in games. For example, an equilibrium in pure strategies always exists [23]. For many games with this property, local search algorithms are known that converge to an equilibrium (e.g., congestion games). However in ASRGs we do not know of any such algorithms.

In this work, we focus on polynomial time algorithms for computing equilibria in ASRGs with general convex costs on parallel edges. Parallel edges are interesting because a number of applications can be modeled using parallel edges, such as load balancing across servers [26], and in traffic models [15]. Further, many results were first obtained for graphs consisting of parallel edges and then extended (e.g., results on the price of collusion [17], extended to series-parallel graphs [5], or on the price of anarchy [19, 7]). These are thus a natural starting point to study equilibrium computation. We believe it likely that some of our structural results extend beyond parallel edges, to nearly-parallel and series-parallel graphs.

Our Contribution. For ASRGs with convex costs on parallel edges, we give two algorithms. Our first algorithm computes an equilibrium[2] in time $O\left((\log|\mathcal{I}|)^n\right)$, where $|\mathcal{I}|$ is the input size and n is the number of players. If the number of players is near-logarithmic in the input size, i.e., $O(\log|\mathcal{I}|/\log\log|\mathcal{I}|)$, this gives a polynomial time algorithm. Our algorithm is based on the idea of reducing equilibrium computation to guessing the marginal costs of the players at equilibrium. The marginal costs turn out to have a number of interesting monotonicity properties, which we use to give a high-dimensional binary search algorithm.

[2] We use the standard notion of polynomial-time computation when outputs are possibly irrational: we say an algorithm is efficient if for any $\epsilon > 0$, the algorithm computes and ϵ-approximate solution in time polynomial in the inputs size and $\log(1/\epsilon)$.

Our second algorithm has running time exponential in the number of edges in the network. If the number of edges is constant, then this gives us a polynomial-time algorithm. Define players to be of the same *type* if they have flow on the same set of edges at equilibrium. The algorithm is based on the following structural result: for parallel edges, computing equilibrium in a general ASRG can be reduced to computing equilibrium in an ASRG where players of the same type also have the same demand. At a high level, this allows us to replace players of the same type by a single player, and then use our previous algorithm. Somewhat surprisingly, this result does not subsume the previous result. This is because the actual partition of players into types is unknown, hence we must enumerate over all possible such partitions, which introduces a factor of $O(n^{|E|})$ to the running time.

Lastly, we show that in general networks, determining existence of a Nash equilibrium where the cost of every player is at most C is NP-hard. Our proof here is a reduction from SUBSET-SUM, and builds upon a construction showing multiplicity of equilibria in ASRGs [4]. Our result parallels early results for bimatrix games [12], which showed that it is NP-hard to determine existence of a Nash equilibrium in bimatrix games where the cost of players is above a threshold [12]. Our proof is computer-assisted, and we use Mathematica to verify properties of equilibria in the games used in our reduction.

The complete proofs can be found in the full version of the paper [6].

Related Work. Existence of equilibria in atomic splitable routing games (and for more general *concave games*) is shown by Rosen [23]. Equilibria in these games are unique when delay functions of the edges are polynomials of degree ≤ 3 [2], when the players are symmetric, or when the underlying network is nearly-parallel [4]. In general, the equilibria may not be unique [4]. For computation, the equilibria can be obtained as the solution to a convex problem if the edge costs are linear, or if the players are symmetric [8]. Huang considers ASRGs with linear delays on a class of networks called *well-designed* which includes series-parallel graphs, and gives a combinatorial algorithm to find the equilibrium [18]. A network is well-designed if for the optimal flow (which minimizes total cost), increasing the total flow value does not decrease the flow on any edge. Recently, Harks and Timmermans gave an algorithm to compute equilibrium for ASRGs with player specific-linear costs on parallel links [16]. This setting allows players to have different cost functions on an edge. Their results use a reduction to integrally splittable flows, where the flow each player puts on an edge is an integral multiple of some quantity. In our case, the equilibrium flow can be irrational, hence these ideas do not seem to work. A number of algorithms for equilibrium computation are also proposed by Marcotte, based on either sequential best-response by the players, or linearization of the cost functions [20]. Marcotte shows conditions under which these algorithms converge to an equilibrium. It is unclear if these algorithms can be shown to run in polynomial time.

Nonatomic games have an infinite set of players, each of which has infinitesimal flow. Unlike ASRGs, equilibria in nonatomic games are well-studied: an equilibrium can be obtained by solving a convex program, and is unique if the costs are strictly increasing [3]. It is also known that ASRGs captures the setting where nonatomic players to form coalitions, and within a coalition players cooperate to minimize the total cost of its flow [17]. A number of papers study the change in total cost as players in a nonatomic game form coalitions, forming an ASRG [17, 5, 18]. Another property of ASRGs that has received a lot of attention is the price of anarchy (PoA), formalised as the ratio of the total cost of the worst equilibrium, to the optimal cost. Upper bounds on the PoA were obtained by Cominetti, Correa, and Stier-Moses [8] and improved upon by Harks [14]. These bounds were shown to be tight [25].

Atomic games where demands are unsplittable are also extensively studied. If all players have the same demand, these are called congestion games. Player strategies may not correspond to paths in a graph; if they do, these are called network congestion games. For congestion games, existence of a potential function is well-known [24], though computing an equilibrium is PLS-hard [10, 1], even if players are symmetric or the edge cost functions are linear. For symmetric players in a network congestion game, or if player strategies correspond to bases of a matroid, the equilibrium can be computed in polynomial time [1, 10].

2 Preliminaries

An *atomic splittable routing game* (ASRG) $\Gamma = (G = (V, E), (v_i, s_i, t_i)_{i\in[n]}, (l_e)_{e\in E}))$ is defined on a directed network $G = (V, E)$ with n players. Each player i wants to send v_i units of flow from s_i to t_i, where v_i is the *demand* of player i. Each edge e has a cost function $l_e(x)$ which is non-negative, increasing, convex and differentiable. Players are indexed so that $v_1 \geq v_2 \geq \ldots \geq v_n$, and the total demand $V := \sum_i v_i$. Vector f^i denotes the flow of player i. By abuse of notation, we say vector f is the flow on the network with n players such that f^i_e denotes the amount of flow player i sends along the edge e, and $f_e := \sum_i f^i_e$ is the total flow on edge e.

Given a flow $f = (f^1, f^2, \ldots, f^n)$ for n players, player i incurs a cost $\mathcal{C}^i_e(f) := f^i_e l_e(f_e)$ on the edge e. His total cost is $\mathcal{C}^i(f) = \sum_{e\in E} \mathcal{C}^i_e(f)$. Each player's objective is to minimize his cost, given the flow of the other players. We say a flow f is at *equilibrium*[3] if no player can unilaterally change his flow and reduce his total cost. More formally,

▶ **Definition 1.** In an ASRG a flow $f = (f^1, f^2, \ldots, f^n)$ is a Nash Equilibrium flow if for every player i and every flow $g = (f^1, f^2, \ldots, f^{i-1}, g^i, f^{i+1}, \ldots, f^n)$, where g^i is a flow of value v_i, $\mathcal{C}^i(f) \leq \mathcal{C}^i(g)$.

The equilibrium flow can be characterized in terms of the marginal costs of each player. Intuitively, the marginal cost for a player on a path is the increase in cost for the player when he increases his flow on the path by a small amount.

▶ **Definition 2.** Given a flow f, the marginal cost for the player i on path p is given by

$$L^i_p(f) = \sum_{e\in p} l_e(f_e) + f^i_e l'_e(f_e)$$

By applying the *Karush-Kuhn-Tucker* conditions [13] for player i's minimization problem, we get the following lemma 3 which characterizes the equilibrium using marginal costs.

▶ **Lemma 3.** *Flow $f = (f^1, f^2, \ldots, f^n)$ is a Nash equilibrium flow iff for any player i and any two directed paths p and q between s_i and t_i such that $f^i_e > 0\ \forall e \in p$, $L^i_p(f) \leq L^i_q(f)$.*

Lemma 3 says that for any player i at equilibrium, the marginal delay $L^i_p(f)$ on all paths p such that $f^i_e > 0\ \forall e \in p$ is equal, and is the minimum over all s_i-t_i paths. In a network of parallel edges, every edge is an s-t path, hence the condition holds at equilibrium with edges replacing paths.

We will frequently use the *support* of a player, where given a flow $f = (f^1, f^2, \ldots, f^n)$, the support of player i, S_i is defined as the set edges with $f^i_e > 0$.

[3] More specifically, a pure Nash equilibrium.

Swamy studies the use of edge tolls to enforce a particular flow as equilibrium [27]. However, if we start with an equilibrium flow, the tolls required are identically zero. The following theorem regarding equilibrium computation is then implicit, and will be useful to us in Section 4.

▶ **Theorem 4** ([27]). *For ASRG Γ, let $(h_e)_{e\in E}$ be the total flow on each edge at an equilibrium. Then given the total flow h, the equilibrium flow for each player can be obtained in polynomial time by solving a convex quadratic program.*

We make the following smoothness assumptions on edge cost functions.

1. Cost functions are continuously differentiable, nonnegative, convex, and increasing.
2. There is a constant $\Psi \geq V$ that satisfies:

$$\Psi \geq \max_{e\in E,\, x\in[0,V]} \left\{ l_e(x),\, l'_e(x),\, l''_e(x),\, \frac{1}{l'_e(x)} \right\}$$

By the first assumption, the edge marginal cost $L^i_e(f)$ is strictly increasing, both with the total flow f_e and player i's flow f^i_e. Define $L^i(f) := \min_{e\in E} L^i_e(f)$. Hence if $\vec{f}$, $\vec{f'}$ are two equilibrium flows, and for some player i and edge e $f_e \geq f'_e$, $f^i_e \geq f'^i_e$, and $f^i_e > 0$, then

$$L^i(f) = L^i_e(f) \geq L^i_e(f') \geq L^i(f')$$

and the second inequality is strict if $f_e > f'_e$ or $f^i_e > f^{i'}_e$. We frequently use this inequality in our proofs.

Also observe that for each edge e and flow values x, $y \in [0, V]$, the following properties of the edge cost functions hold.

$$|x-y| \;\leq\; \delta \Rightarrow\; |l_e(x) - l_e(y)| \leq \delta\Psi \quad \text{and} \quad |l_e(x) - l_e(y)| \;\leq\; \delta \;\Rightarrow\; |x-y| \;\leq\; \delta\Psi$$

3 An Algorithm with Complexity Exponential in the Number of Players

To convey the main ideas of the algorithm, we will ignore issues regarding finite precision computation in this section. In particular, we assume the algorithm carries out binary search to infinite precision, and show that such an algorithm computes the exact equilibrium. In this article we will give the algorithms for computing equilibria assuming that we can compute values upto arbitrary precision. In the full version of the paper [6], we give an implementation of the algorithm with finite precision. We show that our implementation computes an ϵ-equilibrium in time $O\left(mn^2 \left(\log(n\Psi/\epsilon)\right)^n\right)$

Recall that the equilibrium in case of parallel edges is unique [22]. We start with an outline of the algorithm. Our first idea is to reduce the problem of equilibrium computation, to finding the *marginal costs* at equilibrium. We give a function GraphFlow that at a high level, given a vector of marginal costs $\vec{M} = (M^1, \ldots M^n)$, returns a vector of demands $\vec{w} = (w_1, \ldots, w_n)$ and a flow vector $\vec{f} = (f^i_e)_{e\in E, i\in[n]}$ so that (1) $\vec{f}$ is the equilibrium flow for the demand vector $\vec{w}$, and (2) for each player i, the marginal cost $L^i(f) = M^i$. That is, the marginal costs for the players at equilibrium are given by the input vector $\vec{M}$. We show that in fact each marginal cost vector $\vec{M}$ maps to a unique (demand, flow) pair that satisfies these conditions. Hence given marginal costs at equilibrium, the function must return the correct demands $(v_i)_{i\in[n]}$, and the required equilibrium flow. Thus, our problem reduces to finding a marginal cost vector $\vec{M}$ for which GraphFlow returns the *correct* demand vector $\vec{v} = (v^1, v^2, \ldots, v^n)$. We say a demand w^i for player i is *correct* if $w^i = v^i$.

Algorithm 1 GraphFlow($\vec{M}$).

Input: Vector $\vec{M} = (M^i)_{i\in[n]}$ of nonnegative real values

Output: Flow $\vec{f} = (f_i(e))_{i\in[n],e\in E}$ and demands $\vec{w} = (w_i)_{i\in[n]}$ so that $w_i = |f^i|$ and $\vec{f}$ is an equilibrium flow for demands $\vec{w}$ with marginal costs $\vec{M}$.

Assume that $M^1 \geq M^2 \geq \cdots \geq M^n$, else renumber the vector components so that this holds.
for each edge $e \in E$ **do**
  $f_e^i = 0$ for each player $i \in [n]$
  if $l_e(0) \geq M^1$ **then**
    $S_e \leftarrow \emptyset$; continue with the next edge
  for $k = 1 \to n$ **do**
    $S = [k]$
    Let x_e be the unique solution to $kl_e(x) + xl'_e(x) = \sum_{i\in S} M^i$ ▷ Since $l_e(x)$ is strictly increasing and convex, the solution is unique
    $f_e^i = \frac{M^i - l_e(x_e)}{l'_e(x_e)}$ for each player $i \in S$ ▷ Note that $\sum_{i\in S} f_e^i = x_e$
    if ($f_e^i \geq 0$ for all $i \in S$) **and** ($k = n$ **or** $M^{k+1} \leq l_e(x_e)$) **then**
      $f_e \leftarrow x_e$, $S_e \leftarrow S$, continue with the next edge
$w_i \leftarrow \sum_e f_e^i$ for each player i; **return** $(\vec{f}, \vec{w})$

Since only the marginal costs and demands matter to us, we can think of GraphFlow as a function from marginal cost vectors to demand vectors. We then give a high-dimensional binary search algorithm that computes the required marginal cost vector. This proceeds in a number of steps. We first show that the function GraphFlow is continuous, and is monotone in a strict sense: if we increase the marginal cost of a player, then the demand for this player increases, and the demand for every other player decreases. This allows us to show in Lemma 6 that given any marginal costs for the first $n-k$ players, there exist marginal costs for the remaining k players so that the demands returned by GraphFlow for these remaining players is correct. Lemma 6 allows us to ignore first $n-k$ players, and focus on the last k players, since no matter what marginal costs we choose for the first $n-k$ players, we can find marginal costs for the last k players that give the correct demand for these players.

The crux of our binary search algorithm is then Lemma 7, which says the following. Suppose we are given two marginal cost vectors $\vec{M}$ and $\vec{M}'$ that differ only in their last k coordinates, and for which the demands of the last $k-1$ players is equal. Thus, $M^i = M^{i'}$ for all players $i < k$, and the demands returned by GraphFlow($\vec{M}$), GraphFlow($\vec{M}'$) are equal for all players $i > k$. Suppose for the kth player, the demand with marginal costs $\vec{M}$ is higher than the demand with marginal costs $\vec{M}'$. Then Lemma 7 says that k's marginal cost in $\vec{M}$ must be higher than in $\vec{M}'$, i.e., $M^k > M^{k'}$. Thus Lemma 7 allows us to give a recursive binary search procedure. For a player k, the procedure fixes a marginal cost M^k, and finds marginal costs for players $i > k$ so that these players have the correct demand. By Lemma 6, we know that such marginal costs exist. With these marginal costs, if the demand for player k is greater than v_k, then by Lemma 7 M^k is too large. We then reduce M^k, and continue.

Algorithm 1 describes the function GraphFlow. The algorithm considers each edge in turn. For an edge e, it tries to find a subset of players $S \subseteq [n]$ and flows f_e^i so that, for all players $i \in S$, $L_e^i(f) = M^i$, and for all players not in S, $f_e^i = 0$ and $M^i \leq L_e^i(f)$. The set S can be obtained in $O(n)$ time by adding players to S in decreasing order of marginal costs

M^i. Given a set S, summing the equalities $L^i_e(f) = M^i$, we get the following equation with variable f_e:

$$|S|\, l_e(f_e) + f_e l'_e(f_e) \;=\; \sum_{i \in S} M^i \,.$$

Noting that the left-hand side is strictly increasing in f_e, we can solve this equation for f_e using binary search. This gives us the total flow on the edge f_e. We can then obtain the flow for each player by solving, for each player $i \in S$, the following equation:

$$f^i_e \;=\; \frac{M^i - l_e(f_e)}{l'_e(f_e)} \,.$$

We set $f^i_e = 0$ for all players not in S. It can be checked that $\sum_{i \in S} f^i_e = f_e$. If $f^i_e \geq 0$ for all players, and $L^i_e(f) = l_e(f_e) \geq M^i$ for all players not in S, we move on to the next edge. Else, we add the next player with lower marginal cost M^i to the set S, and recompute f_e.

We first establish in Claims 1, 2, and 3 that the algorithm is correct, and gives a continuous map from marginal cost vectors to demand vectors.

▶ **Claim 1.** *Given $\vec{M}$, assume w.l.o.g. that $M^1 \geq M^2 \geq \cdots \geq M^n$. Then GraphFlow($\vec{M}$) returns flow $\vec{f}$ and demands $\vec{w}$ so that, on each edge e and for each player i,*
1. *if $f^i_e = 0$ then $L^i_e(f) \geq M^i$*
2. *if $f^i_e > 0$ then $L^i_e(f) = M^i$*

Thus, $\vec{f}$ is an equilibrium flow for values $\vec{w}$, and if $w_i > 0$ then $M^i = L^i(f)$.

▶ **Claim 2.** *For each vector $\vec{M}$ of marginal costs, there is a unique pair of vectors $(\vec{w}, \vec{f})$ so that:*
1. *$\vec{f}$ is the equilibrium flow for demands $\vec{w}$, and*
2. *for each player i, $L^i(f) \geq M^i$. If $w_i > 0$, then $L^i(f) = M^i$.*

▶ **Corollary 5.** *For a demand vector $\vec{w}$, let $\vec{f}$ be the equilibrium flow, and $M^i = L^i(f)$ be the marginal costs of the players at equilibrium. Then the function GraphFlow($M^1, \ldots, M^n$) returns $(\vec{w}, \vec{f})$ as the output.*

▶ **Claim 3.** *Given marginal costs $\vec{M}$, $\vec{M'}$ so that for player l, $|M^l - M^{l'}| \leq \epsilon$, and $M^j = M^{j'}$ for all players $j \neq l$, let $(\vec{f}, \vec{w})$ and $(\vec{f'}, \vec{w'})$ be the flows and demands returned by GraphFlow. Then for each player i, $|f^i_e - f^{i'}_e| \leq \epsilon'$, where $\epsilon' = 2n\Psi\epsilon$. Hence, for each player i, $|w_i - w'_i| \leq m\epsilon'$.*

Claim 3 then shows that the function GraphFlow is continuous.

In the remainder of the discussion, given a vector of marginal costs $\vec{M}$, we will primarily be concerned with the demands $\vec{w}$ returned by the function GraphFlow. We therefore define the functions GraphVal_i for each player $i \in [n]$. Function GraphVal_i takes as input a vector $\vec{M}$ of marginal costs for the players, and returns the demand w_i, the ith component of the demand vector $\vec{w}$ returned by GraphFlow($\vec{M}$). Claim 4 now shows that the function GraphFlow is monotone: if we increase the input marginal cost of a player, that player's demand goes up, while the demand for all the other players goes down. This is crucial in establishing existence of marginal costs for a subset of players (Lemma 6), and in our binary search algorithm later on.

▶ **Claim 4.** *Consider marginal cost vectors $\vec{M}$ and $\vec{M'}$ that differ only in their k^{th} coordinate, so that $\vec{M} = (M^i)_{i \in [n]}$ and $\vec{M'} = (M^1, M^2, \ldots, M^{k'}, \ldots, M^n)$. For each player i, let $w_i = GraphVal_i(\vec{M})$, and $w'_i = GraphVal_i(\vec{M'})$. If $M^{k'} > M^k$, then the following hold true as well:*

1. $w'_k \geq w_k$,
2. *if* $w'_k > 0$, *then* $w'_k > w_k$,
3. $w'_i \leq w_i$ *for* $i \neq k$, *and*
4. *for any subset of players* P *containing player* k, $\sum_{i \in P} w'_i \geq \sum_{i \in P} w_i$.

Consider the game with cost functions as in Γ, but with n players, each with demand V. Since this is a symmetric game, the equilibrium flow can be computed in polynomial time [8]. We define Λ to be the marginal cost of each player at this equilibrium.

▶ **Lemma 6.** *Let* $S \subseteq [n]$ *be a subset of the players. Given strictly positive input demands* $\hat{w}_i \leq V$ *for players* $i \in S$ *and marginal costs* $M^i \leq \Lambda$ *for players* $i \notin S$, *there exist marginal costs* $\hat{M}^i \leq \Lambda$ *for the players in* S *so that, given input* $((\hat{M}^i)_{i \in S}, (M^i)_{i \notin S})$, *GraphVal*$_i$ *returns* $\hat{w}_i$ *as the demand for players* $i \in S$.

▶ **Lemma 7.** *Given a player* k, *and two marginal cost vectors* $\vec{M}$ *and* $\vec{M}'$ *that satisfy the following properties:*
1. *for all players* $i < k$, $M^i = M^{i'}$,
2. *for all players* $i > k$, *GraphVal*$_i(\vec{M}) =$ *GraphVal*$_i(\vec{M}')$.

Let $w_k =$ *GraphVal*$_k(\vec{M})$, *and* $w'_k =$ *GraphVal*$_k(\vec{M}')$. *If* $w_k > w'_k$, *then* $M^k > M^{k'}$.

Proof. Let $M^k \leq M^{k'}$, and let P be the set of players $\{i \geq k : M^i \leq M^{i'}\}$. Thus $k \in P$. We will show that $w_k \leq w'_k$. The proof proceeds by changing the marginal cost of each player in order from M^i to $M^{i'}$, and considering the effect on total demand of players in P. We show that in this process, the total demand of these players does not increase, and hence

$$\sum_{i \in P} \text{GraphVal}_i(\vec{M}) \;\leq\; \sum_{i \in P} \text{GraphVal}_i(\vec{M}')\,. \tag{1}$$

The expression on the left equals $w_k + \sum_{i \in P, i \neq k} w_i$, while the expression on the right equals $w'_k + \sum_{i \in P, i \neq k} w_i$ since for players $i > k$, $\text{GraphVal}_i(\vec{M}) = \text{GraphVal}_i(\vec{M}')$. Hence, this will show that $w_k \leq w'_k$, as required.
We now need to prove (1). Our proof uses Claim 4. Formally, let $\vec{M}(t) = ((M^{i'})_{i \leq t}, (M^i)_{i > t})$. Then $\vec{M}(0) = \vec{M}$, and $\vec{M}(n) = \vec{M}'$. We will show that

$$\sum_{i \in P} \text{GraphVal}_i(\vec{M}(t)) \;\leq\; \sum_{i \in P} \text{GraphVal}_i(\vec{M}(t+1))\,. \tag{2}$$

For each player $t \in [n]$, there are three cases: either (1) $M^t = M^{t'}$, (2) $M^t \leq M^{t'}$ and $t \in P$, or (3) $M^t > M^{t'}$ and $t \notin P$. We consider these three cases separately. In the first case, $\vec{M}(t) = \vec{M}(t+1)$, and (2) clearly holds. In the second case, by Claim 4 and since $t \in P$, (2) holds. In the third case, again by Claim 4, for all $i \neq t$ the demand either increases or remains the same. Since $t \notin P$, the total demand of players in P either increases or remains the same, and hence (2) holds. This completes the proof. ◀

We now give our algorithm for obtaining the "correct" marginal costs $\vec{M}$, so that $\text{GraphFlow}(\vec{M})$ returns the required equilibrium flow. The algorithm is recursive. For each player k, it picks a candidate marginal cost M^k, and then recursively calls itself to find marginal costs M^i for players $i > k$ so that the demands for these players $i > k$ is correct. If the demand for player k itself is too large, it reduces M^k, and otherwise increases M^k. Thus the algorithm conducts a binary search to find the correct marginal cost for player k, and in each iteration calls itself to determine correct marginal values for players $i > k$.

Algorithm 2 $\text{EqMCost}_k((M^1, \ldots, M^{k-1}))$.

Input: Vector $(M^1, \ldots, M^{k-1})$, with each component $M^i \in [0, \Lambda]$ ▷ If $k = 1$, there is no input required.

Output: Vector $(\vec{M})$ of marginal costs so that the first $k-1$ marginal costs are equal to the inputs, and for players $i \geq k$, the demand $\text{GraphVal}_i(\vec{M}) = v_i$.

1: **if** $k = n$ **then**
2: Using binary search in $[0, \Lambda]$, find M so that $\text{GraphVal}_n((M^i)_{i<n}, M) = v_n$. **return** M.
3: $\text{Low} \leftarrow 0$, $\text{High} \leftarrow \Lambda$, $\text{Mid} \leftarrow (\text{Low} + \text{High})/2$
4: $(M^{k+1}, \ldots, M^n) \leftarrow \text{EqMCost}_{k+1}(M^1, \ldots, M^{k-1}, \text{Mid})$
▷ Call EqMCost_{k+1} to get marginal costs for the remaining player $k+1, \ldots, n$ so that the demand for these players is correct
5: **if** $(\text{GraphVal}_k((M^i)_{i<k}, \text{Mid}, (M^i)_{i>k}) = v_k)$ **then**
6: **return** $(\text{Mid}, (M^i)_{i>k})$
7: **else if** $(\text{GraphVal}_k((M^i)_{i<k}, \text{Mid}, (M^i)_{i>k}) > v_k)$ **then**
8: $\text{High} \leftarrow \text{Mid}$, $\text{Mid} \leftarrow (\text{Low} + \text{High})/2$, goto 4
9: **else if** $(\text{GraphVal}_k((M^i)_{i<k}, \text{Mid}, (M^i)_{i>k}) < v_k)$ **then**
10: $\text{Low} \leftarrow \text{Mid}$, $\text{Mid} \leftarrow (\text{Low} + \text{High})/2$, goto 4

▶ **Theorem 8.** *For ASRG Γ, EqMCost_1 returns marginal cost vector $\vec{M}$ such that $\text{GraphFlow}(\vec{M}) = (\vec{w}, \vec{f})$, where $\vec{w} = \vec{v}$ and $\vec{f}$ is the equilibrium flow in Γ.*

The main ingredient in the proof of Theorem 8 is the following lemma, which shows that recursively, for any player k, the function EqMCost returns correct marginal costs.

▶ **Lemma 9.** *For any vector $(M^1, \ldots, M^{k-1})$ with each component in $[0, \Lambda]$, the function $\text{EqMCost}_k((M^1, \ldots, M^{k-1}))$ returns marginal costs $(M^k, \ldots, M^n)$ for the remaining players so that, for each player $i \geq k$, $\text{GraphVal}_i(M^1, \ldots, M^n) = v_i$.*

Proof. The proof is by induction on n. In the base case, $k = n$, and the input is the vector $(M^1, \ldots, M^{n-1})$ with each component in $[0, \Lambda]$. By Lemma 6, there exists $\hat{M}$ so that $\text{GraphVal}_n(M^1, \ldots, M^{n-1}, \hat{M}) = v_n$. We now show that the value $\hat{M}$ can correctly be found by binary search. Initially, the search interval is $[0, \Lambda]$, and by Lemma 6, $\hat{M}$ lies in the search interval. Assume in some iteration the search interval is $[\text{Low}, \text{High}]$; $\hat{M}$ lies in the search interval; and that $\text{GraphVal}_n(M^1, \ldots, M^{n-1}, \text{Mid}) > v_n$. Since $v_n > 0$, and $v_n = \text{GraphVal}_n(M^1, \ldots, M^{n-1}, \hat{M}) < \text{GraphVal}_n(M^1, \ldots, M^{n-1}, \text{Mid})$, it follows by Lemma 7 that $\hat{M} < \text{Mid}$. Hence, $\hat{M}$ lies in the interval $[\text{Low}, \text{Mid}]$, and we can restrict our search to this space, which is exactly how the binary search proceeds. The case when $\text{GraphVal}_n(M^1, \ldots, M^{n-1}, \text{Mid}) < v_n$ is similar, and $\hat{M}$ then lies in the interval $[\text{Mid}, \text{High}]$.

For the inductive step, we are given player $k < n$. We assume that given any input vector $(M^1, \ldots, M^k)$ with each component in $[0, \Lambda]$, EqMCost_{k+1} returns marginal costs $(M^{k+1}, \ldots, M^n)$ for the remaining players so that for each of these remaining players $i \geq k+1$, $\text{GraphVal}_i(M^1, \ldots, M^n) = v_i$. We need to show that given any input marginal costs $(M^1, \ldots, M^{k-1})$ for the first $k-1$ players, EqMCost_k finds marginal costs $(M^k, \ldots, M^n)$ for players k onwards so that the demand returned for these players $i \geq k$ by GraphVal_i is v_i. Firstly, by Lemma 6, choosing $S = [k, \ldots, n]$ and $\hat{w}_i = v_i$ for players $i \in S$, there exist marginal costs $(\hat{M}^k, \ldots, \hat{M}^n)$ so that for all $i \geq k$, $\text{GraphVal}_i((M^i)_{i<k}, (\hat{M}^i)_{i \geq k}) = v_i$. We now show that the binary search procedure in EqMCost_k finds the required marginal cost $\hat{M}^k$. By Lemma 6, $\hat{M}^k$ lies in the initial search interval $[0, \Lambda]$. Assume that in some iteration,

$\hat{M}^k$ lies in the search interval [Low, High], and Mid = (Low + High)/2. By the induction hypothesis, $\text{EqMCost}_{k+1}(M^1, \ldots, M^{k-1}, \text{Mid})$ returns marginal costs $(M^{k+1}, \ldots, M^n)$ for the players $k+1, \ldots, n$ so that for each of these players $i \geq k+1$ (but not player k), $\text{GraphVal}_i((M^i)_{i<k}, \text{Mid}, (M^i)_{i>k}) = v_i$. Further, for each player $i \geq k$, $\text{GraphVal}_i((M^i)_{i<k}, (\hat{M}^i)_{i\geq k}) = v_i$. Suppose that for player k, $\text{GraphVal}_k((M^i)_{i<k}, \text{Mid}, (M^i)_{i>k}) > v_k = \text{GraphVal}_i((M^i)_{i<k}, (\hat{M}^i)_{i\geq k})$. Then by Lemma 7, Mid $> \hat{M}^k$, and hence $\hat{M}^k$ lies in the interval [Mid, High]. The algorithm then reduces the search space to this interval, and continues. If $\text{GraphVal}_k((M^i)_{i<k}, \text{Mid}, (M^i)_{i>k}) < v_k$, it can be similarly shown that Mid $< \hat{M}^k$. Thus, $\hat{M}^k$ always lies in the search space [Low, High], which is halved in each iteration. The binary search is thus correct, and must eventually terminate. ◀

Proof of Theorem 8. By Lemma 9, EqMCost_1 returns a marginal cost vector $\vec{M} = (M^1, \ldots, M^n)$ so that $\text{GraphVal}_i(\vec{M}) = v_i$ for each player i. Let $\vec{v} = (v_1, \ldots, v_n)$. By definition of the function GraphVal, this implies that $\text{GraphFlow}(\vec{M})$ returns vectors $\vec{v}$ and $\vec{f}$. Finally by Claims 1 and 2, $\vec{f}$ is the equilibrium flow for demands $\vec{v}$, as required. ◀

Implementation and Complexity. Under the assumption that binary search could be done to arbitrary precision, we showed that the algorithm EqMCost is correct. However, the solutions to the polynomial equations could be irrationals, and thus the algorithm given is not a finite algorithm. In the full version of paper [6], we show that for any given error parameter ϵ, we can implement EqMCost to run in time $O\left(\text{poly}(\log \Psi, \log \frac{1}{\epsilon}, m, n)\right)$, and return an ϵ-equilibria. We say a flow $\vec{f}$ is an ϵ-equilibrium if any player i has flow only on ϵ-minimum marginal cost edges. That is, if $f_e^i > 0$ for player i on edge e, then $L_e^i(f) \leq \min_{e'} L_{e'}^i(f) + \epsilon$. Conventionally, a strategy profile is an ϵ-equilibrium if no player can improve it's cost by ϵ. One can check that the two are equivalent: if a flow f is an ϵ-equilibrium by our definition, then no player i can improve its cost by more than ϵv_i, where v_i is its demand.

The work in giving an implementation for the algorithm EqMCost is in implementing the binary search correctly, up to some error parameter ϵ. Since the algorithm is iterative, this error grows across each iteration, and bounding the error in each iteration is quite technical. Additionally, approximate versions of some of the results for EqMCost have to be reproved. Further details are given in the full version of paper [6].

4 An Algorithm with Complexity Exponential in Number of Edges

Our second algorithm is based on Theorem 10, which shows that at equilibrium the supports of players form chains.

▶ **Theorem 10** ([4]). *Consider an ASRG with n players on a graph consisting of parallel edges*[4]*, and let f be the equilibrium flow. Then $L^1(f) \geq \cdots \geq L^n(f)$. Consequently, the supports $S_1(f) \supseteq \cdots \supseteq S_n(f)$.*

Thus in an ASRG on m parallel links, there exist numbers $1 = a_1 < a_2 < \cdots < a_T \leq n$ with $T \leq m$, so that players with indices in $[a_i, a_{i+1} - 1]$ have the same support at equilibrium. Define a *type set* $\mathcal{T} = (P_1, \ldots, P_T)$ with $T \leq m$ to be a partition of the players so that players in a set P_t in the partition have consecutive indices. Hence, a type set $\mathcal{T} = (P_1, \ldots, P_T)$ can be denoted by a sequence of numbers $(a_1, \ldots, a_T)$ where $1 = a_1 < a_2 < \cdots < a_T \leq n$ and P_t consists of the players with indices $a_t, \ldots, a_{t+1} - 1$. We say a type set is *valid* for

[4] The proof by Bhaskar et al. [4] is for series-parallel graphs which are a superset of parallel link graphs.

Γ iff two players in the same partition in $\mathcal{T}$ also have the same support in the equilibrium. Theorem 10 then shows that in a graph consisting of m parallel links, there is a type set that is valid.

We will now give an algorithm with running time that is exponential in the number of edges, using the algorithm from Section 3, which is exponential in the number of players, and Theorem 4. Our algorithm in this section crucially uses Lemma 11, which has the following content. Let $\mathcal{T} = (P_1, \ldots, P_T)$ be a type set, not necessarily valid, for a game Γ. Consider the game $\Gamma^{\mathcal{T}}$ where for each set $P_t \in \mathcal{T}$, we replace the players in P_t with $|P_t|$ players that have the same demand, given by $\sum_{i \in P_t} v_i/|P_t|$. That is, we pick a set P_t, and replace all players in this set by players with demands equal to the average demand of players in P_t. We do this for each set P_t. Lemma 11 then says that if $\mathcal{T}$ is valid for Γ, then the total flow on any edge does not change between Γ and $\Gamma^{\mathcal{T}}$.

▶ **Lemma 11.** *Let $\mathcal{T} = (P_1, \ldots, P_T)$ be the valid type set for game Γ with n players on a network of parallel edges. Let f and g be the respective equilibrium flows for games Γ and $\Gamma^{\mathcal{T}}$ respectively. Then on each edge e, $f_e = g_e$.*

The proof of Lemma 11 closely follows an earlier proof of uniqueness of equilibrium in an ASRG [4, Theorem 5]. Lemma 11 implies that in an ASRG on parallel edges we can replace players that have the same support with players that have the same demand without affecting the total flow at equilibrium on the edges. Further, given the total flow on each edge at equilibrium in a general network, the flow for each player can be computed by Theorem 4.

The algorithm EqMCost runs a binary search for the marginal cost at equilibrium for each player. For each player i, EqMCost$_i$ runs a binary search, in each iteration of which it calls EqMCost$_{i+1}$. If each binary search runs for R iterations, and each iteration takes time S, then this recursively gives a running time of $O(S\,R^n)$. However, if players i, $i+1$ have the same demand, then by Theorem 10 they also have the same marginal cost at equilibrium. Hence we can run the binary search for the marginal cost at equilibrium, for both players *simultaneously*. This is the basic idea for the modification. For a given type set $\mathcal{T} = (P_1, \ldots, P_T)$, and the game $\Gamma^{\mathcal{T}}$, we know that all players in the same set P_t have the same marginal cost at equilibrium. Hence, we run the binary search once for each set P_t, rather than once for each player. By Theorem 10, the number of types $T \leq m$.

This gives us the reduced running time of $O(S\,R^m)$ for type set $\mathcal{T}$, for each run of the modified algorithm EqMCostTy. However, note that we run the algorithm once for each possible type set. If the game is played on m parallel links, then the number of possible type sets is about $(n+m)^m$. In the full version of paper [6], we show that this takes a running time of $O\left((n+m)^m m^3 \left(\log(n\Psi/\epsilon)\right)^m\right)$.

5 Hardness of Computing Equilibria

Prior to the proof of PPAD-hardness of computing a (mixed) Nash equilibrium in bimatrix games, Gilboa and Zemel showed that it was NP-hard to determine if there existed an equilibrium in bimatrix games where every player had payoff above a given threshold C [12]. We show a similar result for ASRGs.

▶ **Theorem 12.** *Given an ASRG with convex, strictly increasing and continuously differentiable cost functions, it is NP-hard to determine if there exists an equilibrium at which cost of every player is at most C.*

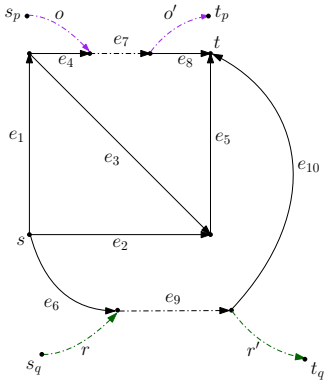

Figure 1 ASRG $\mathcal{G}$, with multiple equilibria.

The main idea of the proof is to build upon the existence of multiple equilibria in ASRGs, and is a reduction from SUBSET-SUM. In this problem, we are given a set $S = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{N}$ such that the sum of elements in S is M, and we want to determine if there exists a subset $T \subseteq S$ such that the sum of elements in T is $M/2$. SUBSET-SUM problem is known to be NP-complete. Our reduction is in two steps. First, we construct an ASRG $\mathcal{G}$ with four players b, r, p, and q, and exactly three equilibria, one of which is irrational, which is used as gadget in the reduction (Figure 1).[5] This construction builds upon an earlier example showing multiplicity of equilibria in ASRGs [4]. We will be mainly concerned with the rational equilibrium flows, say f and g. We choose the cost functions so that (i) $\mathcal{C}^p_{e_7}(g) > \mathcal{C}^p_{e_7}(f)$, and (ii) the sum of costs of players p and q are equal for f and g, that is,

$$\Lambda \;:=\; \mathcal{C}^p_{e_7}(f) + \mathcal{C}^q_{e_9}(f) \;=\; \mathcal{C}^p_{e_7}(g) + \mathcal{C}^q_{e_9}(g) \tag{3}$$

Then $\mathcal{C}^q_{e_9}(f) > \mathcal{C}^q_{e_9}(g)$.

We now repeat this subgame n times in series, once for each element in the set S in the SUBSET-SUM instance. Each subgame is independent of the others, i.e., the players b_i and r_i in i^{th} subgame are local to that subgame and do not play any role in other subgames. All the subgames are connected by players p and q, who can only use one edge (e_7 and e_9 respectively) in each subgame. We show that f, g, and h continue to be the only equilibria within each subgame. In the ith subgame $\mathcal{G}_i$, we multiply each cost function by s_i. This causes all costs to get multiplied by s_i, and does not affect the equilibria. Thus, in each subgame $\mathcal{G}_i$, player p has costs $s_i\mathcal{C}^p_{e_7}(f)$ and $s_i\mathcal{C}^p_{e_9}(g)$ in equilibrium flows f, g, and h (confined to the subgame) respectively. Similar for player q. Roughly, we think of equilibrium f in a subgame as putting s_i in the subset S, and equilibrium g as leaving s_i out.

We will show that if the given instance satisfies SUBSET-SUM, then there exists an equilibrium at which both players p and q have cost $(M\Lambda)/2$, otherwise at least one of them has cost strictly greater than $(M\Lambda)/2$. At equilibrium in the game, let F be the subgames where f is the equilibrium, and G be the subgames where g is the equilibrium. Then the total cost of players p and q is

$$\mathcal{C}^p_{e_7}(f)\sum_{i\in F} s_i + \mathcal{C}^p_{e_7}(g)\sum_{i\in G} s_i + \mathcal{C}^q_{e_9}(f)\sum_{i\in F} s_i + \mathcal{C}^q_{e_9}(g)\sum_{i\in G} s_i \;=\; M\Lambda$$

[5] We use Mathematica to verify properties of equilibria in the games used in the reduction. Further details are given in the full version of paper [6].

where the equality follows from (3). From $\mathcal{C}^p_{e_7}(g) > \mathcal{C}^p_{e_7}(f)$, it follows that the cost of each player p, q is $(M\Lambda)/2$ iff at equilibrium, $\sum_{i\in F} s_i = \sum_{i\in G} s_i$. Else, since the sum of costs of the two players is constant, exactly one player has cost above $(M\Lambda)/2$.

To complete the proof, we add player-specific edges to ensure that $(M\Lambda)/2$ is large enough so that all the other players b_i, r_i always have cost at most $(M\Lambda)/2$ at any equilibrium.

References

1. Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 613–622, 2006.
2. Eitan Altman, Tamer Başar, Tania Jimenez, and Nahum Shimkin. Competitive routing in networks with polynomial costs. *Automatic Control, IEEE Transactions on*, 47(1):92–96, 2002.
3. MJ Beckmann, CB Mc Guire, and CB Weinstein. Studies in the economics of transportation, Yale University Press. *New Haven, Connecticut, USA*, 1956.
4. Umang Bhaskar, Lisa Fleischer, Darrell Hoy, and Chien-Chung Huang. On the uniqueness of equilibrium in atomic splittable routing games. *Math. Oper. Res.*, 40(3):634–654, 2015. doi:10.1287/moor.2014.0688.
5. Umang Bhaskar, Lisa Fleischer, and Chien-Chung Huang. The price of collusion in series-parallel networks. In *Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings*, pages 313–326, 2010.
6. Umang Bhaskar and Phani Raj Lolakapuri. Equilibrium computation in atomic splittable routing games with convex cost functions. *CoRR*, abs/1804.10044, 2018. arXiv:1804.10044.
7. Kshipra Bhawalkar, Martin Gairing, and Tim Roughgarden. Weighted congestion games: The price of anarchy, universal worst-case examples, and tightness. *ACM Trans. Economics and Comput.*, 2(4):14:1–14:23, 2014.
8. Roberto Cominetti, José R. Correa, and Nicolás E. Stier Moses. The impact of oligopolistic competition in networks. *Operations Research*, 57(6):1421–1437, 2009. doi:10.1287/opre.1080.0653.
9. Constantinos Daskalakis. On the complexity of approximating a Nash equilibrium. *ACM Trans. Algorithms*, 9(3):23:1–23:35, 2013.
10. Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 604–612, 2004.
11. Jugal Garg, Ruta Mehta, Vijay V. Vazirani, and Sadra Yazdanbod. Settling the complexity of Leontief and PLC exchange markets under exact and approximate equilibria. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 890–901, 2017.
12. Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1(1):80–93, 1989.
13. Geoff Gordon and Ryan Tibshirani. Karush-Kuhn-Tucker conditions. *Optimization*, 10(725/36):725, 2012.
14. Tobias Harks. Stackelberg strategies and collusion in network games with splittable flow. *Theory of Computing Systems*, 48(4):781–802, 2011.
15. Tobias Harks, Ingo Kleinert, Max Klimm, and Rolf H Möhring. Computing network tolls with support constraints. *Networks*, 65(3):262–285, 2015.

16 Tobias Harks and Veerle Timmermans. Equilibrium computation in atomic splittable singleton congestion games. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 442–454, 2017.

17 Ara Hayrapetyan, Éva Tardos, and Tom Wexler. The effect of collusion in congestion games. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 89–98, 2006.

18 Chien-Chung Huang. Collusion in atomic splittable routing games. *Theory Comput. Syst.*, 52(4):763–801, 2013. doi:10.1007/s00224-012-9421-4.

19 Elias Koutsoupias and Christos H. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

20 Patrice Marcotte. Algorithms for the network oligopoly problem. *Journal of the Operational Research Society*, pages 1051–1065, 1987.

21 Ariel Orda, Raphael Rom, and Nahum Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking (ToN)*, 1(5):510–521, 1993.

22 Oran Richman and Nahum Shimkin. Topological uniqueness of the Nash equilibrium for selfish routing with atomic users. *Math. Oper. Res.*, 32(1):215–232, 2007. doi:10.1287/moor.1060.0229.

23 J Ben Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534, 1965.

24 Robert W Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

25 Tim Roughgarden and Florian Schoppmann. Local smoothness and the price of anarchy in splittable congestion games. *Journal of Economic Theory*, 156:317–342, 2015.

26 Subhash Suri, Csaba D Tóth, and Yunhong Zhou. Selfish load balancing and atomic congestion games. *Algorithmica*, 47(1):79–96, 2007.

27 Chaitanya Swamy. The effectiveness of Stackelberg strategies and tolls for network congestion games. *ACM Trans. Algorithms*, 8(4):36:1–36:19, 2012. doi:10.1145/2344422.2344426.