Efficient Methods for Selfish Network Design*

Dimitris Fotakis[1], Alexis C. Kaporis[2,3], and Paul G. Spirakis[2,3]

[1] School of Electrical and Computer Engineering
National Technical University of Athens, 15780 Athens, Greece
[2] Department of Computer Engineering and Informatics
University of Patras, 26500 Patras, Greece
[3] Research Academic Computer Technology Institute
N. Kazantzaki Str., University Campus, 26500 Patras, Greece
fotakis@cs.ntua.gr, kaporis@ceid.upatras.gr, spirakis@cti.gr

Abstract. Intuitively, Braess's paradox states that *destroying* a part of a network may *improve* the common latency of selfish flows at Nash equilibrium. Such a paradox is a pervasive phenomenon in real-world networks. Any administrator, who wants to improve equilibrium delays in selfish networks, is facing some basic questions: (i) Is the network *paradox-ridden*? (ii) How can we delete some edges to *optimize* equilibrium flow delays? (iii) How can we modify edge latencies to *optimize* equilibrium flow delays?

Unfortunately, such questions lead to NP-hard problems in general. In this work, we impose some natural restrictions on our networks, e.g. we assume strictly increasing linear latencies. Our target is to formulate *efficient algorithms* for the three questions above. We manage to provide:

- A polynomial-time algorithm that decides if a network is paradox-ridden, when latencies are linear and strictly increasing.
- A reduction of the problem of deciding if a network with arbitrary linear latencies is paradox-ridden to the problem of generating all optimal basic feasible solutions of a Linear Program that describes the optimal traffic allocations to the edges with constant latency.
- An algorithm for finding a subnetwork that is almost optimal wrt equilibrium latency. Our algorithm is *subexponential* when the number of paths is polynomial and each path is of polylogarithmic length.
- A polynomial-time algorithm for the problem of finding the best subnetwork, which outperforms any known approximation algorithm for the case of strictly increasing linear latencies.
- A polynomial-time method that turns the optimal flow into a Nash flow by deleting the edges not used by the optimal flow, and performing minimal modifications to the latencies of the remaining ones.

Our results provide a deeper understanding of the computational complexity of recognizing the Braess's paradox most severe manifestations, and our techniques show novel ways of using the probabilistic method and of exploiting convex separable quadratic programs.

* The 3rd author was partially supported by the FET Unit of EC, under contract no. FP6-021235-2 (ARRIVAL), and by the ICT Programme of the EU, under contract no. ICT-2008-215270 (FRONTS). Part of this work was done while the 1st author was with the Dept. of Information and Communication Systems Eng., University of the Aegean, Greece.

1 Introduction

A typical instance of *selfish routing* consists of a directed network with a source s and a destination t, with each edge having a non-decreasing function that determines the edge's latency as a function of its traffic, and a rate of traffic divided among an infinite population of players, each willing to route a negligible amount of traffic through a $s-t$ path. The players seek to minimize the sum of edge latencies on their path. Observing the traffic caused by others, each player selects a $s-t$ path of minimum latency. Thus, they reach a *Nash equilibrium* (aka a *Wardrop equilibrium*), where all players route their traffic on paths of equal minimum latency. Under some general assumptions on the latency functions, a Nash equilibrium flow (or simply a *Nash flow*) exists and the common (and the total) players' latency in a Nash flow is unique (see e.g. [25,28]).

Motivation and Previous Work. A Nash equilibrium may not optimize the network performance, measured by the *total latency* incurred by all players. The main tool for quantifying and understanding the performance degradation due to the players' non-cooperative and selfish behaviour has been the *Price of Anarchy* (PoA), which was suggested in a groundbreaking work by Koutsoupias and Papadimitriou [17]. The PoA is the ratio of the total latency of the Nash flow to the optimal total latency. Roughgarden [26] proved that the PoA is independent of the network topology and at most $\rho(\mathcal{D})$, where ρ only depends on the class of latency functions $\mathcal{D}$ (e.g. ρ is 4/3 for linear latencies).

With the PoA very well understood, a few natural approaches for reducing it have been investigated. A simple approach that does not require any network modifications is *Stackelberg routing* [16], where the administrator exploits a small fraction of coordinated traffic to improve the quality of the Nash flow reached by the remaining selfish traffic. For parallel-link networks with arbitrary latencies and for general networks with polynomial latencies, the coordinated traffic can be allocated so that the PoA decreases smoothly to 1 as the fraction of the coordinated traffic increases (see e.g. [27,15,4], and [9] for the case of atomic players with unsplittable traffic). Unfortunately, there are instances for which the PoA remains unbounded under any allocation of the coordinated traffic [4], and instances where enforcing the optimal flow requires a large fraction of coordinated traffic [13]. A different approach is to introduce edge-dependent per-unit-of-traffic *tolls*, that influence the players' selfish choices and induce the optimal flow as the Nash flow of the modified instance. In the *refundable tolls* setting, where tolls affect the players' cost but not the network performance, tolls that enforce the optimal flow can be computed efficiently even if the players have different latency-vs-tolls valuations (see e.g. [7,8,14], see also [6,10] on the performance of refundable tolls for atomic players). However, the idea of tolls is not appealing to the players, since large tolls that significantly increase the players' disutility may be required to enforce the optimal flow (see e.g. [8]).

A simpler way of improving network performance at equilibrium is to exploit the essence of the Braess's paradox [5], namely that removing some network edges may decrease the latency of the Nash flow (see Fig. 1 for an example). Thus, given

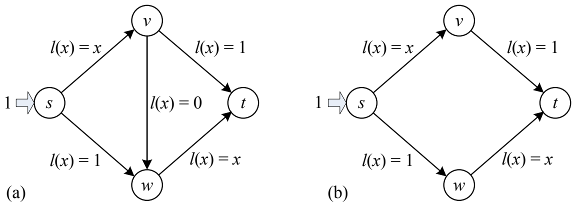

Fig. 1. (a). The optimal flow routes 1/2 unit of traffic on the upper path (s, v, t) and 1/2 unit on the lower path (s, w, t), and achieves a total latency of 3/2. In the Nash flow, all traffic goes through the path (s, v, w, t). The players' latency is 2, and the PoA is 4/3. (b). Without the edge (v, w), the Nash flow coincides with the optimal flow. The network (a) is *paradox-ridden*, and the network (b) is its *best subnetwork*.

an instance of selfish routing, the administrator seeks for the *best subnetwork*, i.e. the subnetwork minimizing the players' latency at equilibrium. Compared to Stackelberg routing and refundable tolls, edge removal is simpler and more appealing. For the administrator, blocking the traffic on some edges is easier and less expensive to implement than setting up a mechanism for collecting tolls on every edge and refunding them to the players. As for the players, edge removal is applied only if it results in a (significant) improvement on their equilibrium latency, which is preferable to either tolls, that increase their disutility, or a Stackelberg strategy, that allocates the coordinated traffic to slower paths.

Recent work indicates that edge removal can improve the performance of real-world networks (see e.g. [28]). In this vein, Valiant and Roughgarden [30] proved that the Braess's paradox occurs with high probability on random networks, and that for a natural distribution of linear latencies, edge removal improves the equilibrium latency by a factor arbitrarily close to 4/3 (i.e. the worst-case PoA for linear latencies) with high probability. Unfortunately, Roughgarden [29] proved that it is NP-hard not only to find the best subnetwork, but also to compute any meaningful approximation to the equilibrium latency on the best subnetwork. In particular, he showed that even for linear latencies, it is NP-hard to distinguish between *paradox-free* instances, where edge removal cannot improve the equilibrium latency, and *paradox-ridden* instances, where the total latency of the Nash flow on the best subnetwork is equal to the optimal total latency (i.e. edge removal can decrease the PoA to 1). This implies that for any $\varepsilon > 0$, it is NP-hard to approximate the equilibrium latency on the best subnetwork within a factor of $4/3 - \varepsilon$ for linear latencies, and within a factor of $\lfloor n/2 \rfloor - \varepsilon$ for general latencies, where n denotes the number of nodes. In fact, the only known algorithm for approximating the equilibrium latency on the best subnetwork is the trivial one, which does not remove any edges and achieves an approximation ratio of 4/3 for linear latencies and $\lfloor n/2 \rfloor$ for general latencies.

Contribution. The motivating question for this work is whether there are some practically interesting settings where a set of edges, whose removal significantly improves the equilibrium latency, can be computed efficiently. Rather surprisingly, we answer this question in the affirmative for several interesting cases. To the best of our knowledge, our results are the first of theoretical nature which indicate that the Braess's paradox can be efficiently detected and eliminated in many interesting cases. Throughout this paper, we mostly focus on the important case of linear latencies, even though some of our results can be generalized to other classes of latency functions (e.g. polynomial latencies).

We first consider the problem of *recognizing paradox-ridden* instances. Even though this problem is NP-complete for arbitrary linear latencies [29], we show that it becomes *polynomially solvable* for the important case of strictly increasing linear latencies[1]. Recognizing a paradox-ridden instance is equivalent to deciding whether the instance admits an optimal flow that is a Nash flow on its subnetwork (cf. Lemma 1). Then removing all edges not used by the optimal flow yields the best subnetwork. However, an instance may admit many different optimal flows. In fact, the NP-hardness proofs in [29] employ instances with exponentially many optimal flows. On the other hand, if the optimal flow is unique, we can recognize paradox-ridden instances by computing it and checking whether it is a Nash flow on its subnetwork. Based on this observation, we present a *polynomial-time* algorithm that recognizes *paradox-ridden* instances with strictly increasing linear latencies (cf. Theorem 1). Furthermore, we reduce the problem of recognizing a paradox-ridden instance with arbitrary linear latencies to the problem of generating all optimal basic feasible solutions of a Linear Program that describes the optimal traffic allocations to the constant latency edges (cf. Theorem 2).

Then we proceed to the more general problem of computing the *best subnetwork* and its equilibrium latency. For instances with polynomially many paths, each of polylogarithmic length, and arbitrary linear latencies, we present a subexponential-time approximation scheme. For any $\varepsilon > 0$, the algorithm computes a subnetwork with an ε-Nash flow in which the players' latencies are within an additive term of $\varepsilon/2$ from the equilibrium latency on the best subnetwork. The running time is exponential in $\text{poly}(\log m)/\varepsilon^2$, where m is the number of edges (cf. Theorem 3). The analysis is based on a novel application of the Probabilistic Method [1] motivated by Althöfer's Lemma [2] and its application to the computation of approximate Nash equilibria for bimatrix games [21,20]. In particular, we apply the Probabilistic Method and show that any flow on any network admits an ε-approximate "sparse" flow, which assigns traffic to $\mathrm{O}(\log m/\varepsilon^2)$ paths (cf. Lemma 2). The proof has to take advantage of the network structure, since the number of paths may be exponential in m. Hence, our result comprises a novel (and more efficient) extension of Althöfer's Lemma to the network setting. In addition, the application to the best subnetwork approximation deals with a congestion game with an infinite number of players, and is fundamentally different from the application of Althöfer's Lemma to approximation of Nash equilibria for bimatrix games. In fact, to the best of our knowledge, this is the first time that similar techniques are applied in the context of selfish routing.

For instances with strictly increasing linear latencies that are not paradox-ridden, we show that there is an instance-dependent $\delta > 0$, such that the

[1] Constant latency edges represent links of practically infinite capacity. Therefore real-world networks are most unlikely to contain many of them, if they contain any.

equilibrium latency is within a factor of $4/3 - \delta$ from the equilibrium latency on the best subnetwork. Since we can efficiently compute the best subnetwork for paradox-ridden instances, we can use the trivial algorithm for the remaining ones, and approximate the equilibrium latency on the best subnetwork within a factor strictly smaller than the inapproximability threshold[2] of 4/3 (cf. Theorem 4).

If the instance is not paradox-ridden however, it is not possible to turn the optimal flow into a Nash flow by just removing edges. Enforcing the optimal flow is possible, if in addition to removing edges, the administrator can modify the latency functions. In Section 5, we present a polynomial-time algorithm for the problem of minimally modifying the latency functions of the edges used by the optimal flow so that the optimal flow is enforced as a Nash flow on the subnetwork used by the optimal flow with the modified latencies (cf. Theorem 5).

Other Related Work. For the problem of finding the best subnetwork in the atomic model, Azar and Epstein [3] obtained strong inapproximability results similar to those in [29]. Interestingly, the Braess's paradox can be dramatically more severe in multi-commodity instances than in single-commodity ones. More precisely, Lin *et al.* [18] proved that for single-commodity instances with general latency functions, the removal of at most k edges cannot improve the equilibrium latency by a factor greater than $k+1$. On the other hand, Lin *et al.* [19] presented a 2-commodity instance where the removal of a single edge improves the equilibrium latency by a factor of $2^{\Omega(n)}$. As for the impact of the network topology, Milchtaich [23] proved that the Braess's paradox does not occur in (single-commodity) series-parallel networks, which is precisely the class of networks that do not contain the network in Fig. 1.a as a topological minor.

2 Model, Preliminaries, and Problem Definitions

A *selfish routing instance* is a tuple $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$, where $G(V,E)$ is a directed network with a source s and a destination t, $\ell_e : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is a non-decreasing latency function associated with each edge e, and $r > 0$ is the rate of traffic entering the network at s and leaving the network at t. Let $n = |V|$ and $m = |E|$, and let $\mathcal{P}$ denote the set of simple $s-t$ paths in G. We assume that the edge latency functions $\ell_e(x)$ are continuous, differentiable, and convex in the interval $[0, r]$. We mostly focus on *linear* latency functions $\ell_e(x) = a_e x + b_e$, with rational coefficients $a_e, b_e \geq 0$. Such a function is *constant* if $a_e = 0$.

Given a selfish routing instance $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$, any subgraph $H(V,E')$, $E' \subseteq E$, obtained from G by edge deletions is called a *subnetwork* of G. H has the same source s and destination t as G, and the edges of H

[2] The reduction of [29, Theorem 3.3] constructs instances where almost all edges have constant latency 0. Using $\ell(x) = \varepsilon x$, for some very small $\varepsilon > 0$, instead of 0, we can show that even for strictly increasing linear latencies, it is NP-hard to approximate the equilibrium latency on the best subnetwork within a factor considerably smaller than 4/3 for all instances. In this sense, our result is best possible.

preserve their latencies in $\mathcal{G}$. Each instance $\mathcal{H} = (H(V, E'), (\ell_e)_{e \in E'}, r)$, where $H(V, E')$ is a subnetwork of $G(V, E)$, is called a *subinstance* of $\mathcal{G}$.

Flows. A ($\mathcal{G}$-feasible) *flow* f is a non-negative vector indexed by $\mathcal{P}$ so that $\sum_{p \in \mathcal{P}} f_p = r$. For a flow f, let $f_e = \sum_{p : e \in p} f_p$ be the amount of flow that f routes on edge e. Flows f and g are *different* if there is an edge e with $f_e \neq g_e$. An edge e is used by flow f if $f_e > 0$. Given a flow f, the latency of each edge e is $\ell_e(f_e)$, and the latency of each path p is $\ell_p(f) = \sum_{e \in p} \ell_e(f_e)$. For an instance $\mathcal{G}$ defined on a network $G(V, E)$ and a flow f, we let $E_f = \{e \in E : f_e > 0\}$ be the set of edges used by f, and $G_f(V, E_f)$ be the subnetwork of G corresponding to f. A flow f is acyclic if G_f contains no cycles.

The *total latency* of a flow f is $C(f) = \sum_{p \in \mathcal{P}} f_p \ell_p(f) = \sum_{e \in E} f_e \ell_e(f_e)$. The *optimal* flow of instance $\mathcal{G}$, denoted o, minimizes the total latency among all $\mathcal{G}$-feasible flows. We let $L^*(\mathcal{G}) = C(o)/r$ be the average latency in the optimal flow. We note that for every subinstance $\mathcal{H}$ of $\mathcal{G}$, $L^*(\mathcal{H}) \geq L^*(\mathcal{G})$. For the latency functions considered in this paper, an optimal flow can be computed efficiently, while for strictly increasing latencies, the optimal flow is unique (in the sense that all optimal flows route the same amount of traffic on every edge).

Nash Flows. The traffic is divided among an infinite population of players, each willing to route a negligible amount of traffic through a minimum latency $s - t$ path. A flow f is a *Nash flow*, if it routes all traffic on minimum latency paths. Formally, f is a Nash flow if for every path p with $f_p > 0$, and every path p', $\ell_p(f) \leq \ell_{p'}(f)$. Therefore, in a Nash flow f, all players incur a common latency $L(f) = \min_{p : f_p > 0} \ell_p(f)$ on their paths, and the total latency is $C(f) = rL(f)$.

For the latency functions considered in this paper, every instance $\mathcal{G}$ admits at least one Nash flow, and the common players' latency (and thus the total latency) is the same for all Nash flows (see e.g. [28]). For instance $\mathcal{G}$, we let $L(\mathcal{G})$ (resp. $rL(\mathcal{G})$) be the common players' latency (resp. total latency) for some Nash flow of $\mathcal{G}$. We refer to $L(\mathcal{G})$ (resp. $rL(\mathcal{G})$) as the equilibrium latency (resp. equilibrium total latency) of $\mathcal{G}$. We note that for every subinstance $\mathcal{H}$ of $\mathcal{G}$, $L^*(\mathcal{G}) \leq L(\mathcal{H})$, and that there may be subinstances $\mathcal{H}$ with $L(\mathcal{H}) < L(\mathcal{G})$ (see e.g. Fig. 1). For the class of latency functions considered in this paper, a Nash flow can be computed efficiently, while for strictly increasing latencies, the Nash flow is unique (see e.g. [28, Cor. 2.6.4]).

ε-Nash flows. The definition of a Nash flow can be naturally generalized to that of an "almost Nash" flow. Formally, for some $\varepsilon > 0$, a flow f is an ε-Nash flow if for every path p with $f_p > 0$, and every path p', $\ell_p(f) \leq \ell_{p'}(f) + \varepsilon$.

Price of Anarchy. The *Price of Anarchy* (PoA) of a selfish routing instance $\mathcal{G}$, denoted $\rho(\mathcal{G})$, is the ratio of the equilibrium total latency to the optimal total latency. By the discussion above, $\rho(\mathcal{G}) = L(\mathcal{G})/L^*(\mathcal{G})$.

Other Notation and Conventions. For any integer $k \geq 1$, we let $[k] = \{1, \ldots, k\}$. For an event E in a sample space, we let $\mathbb{P}[E]$ denote the probability of event E happening. For a random variable X, we let $\mathbb{E}[X]$ denote the *expectation* of X. For convenience and wlog., we normalize the traffic rate to 1.

Then $L(\mathcal{G})$ equals both the common players' latency and the total latency at equilibrium, and $L^*(\mathcal{G})$ equals both the optimal average latency and the optimal total latency of $\mathcal{G}$. With the traffic rate normalized to 1, we sometimes identify a selfish routing instance with the corresponding network.

Paradox-Free and Paradox-Ridden Instances. An instance $\mathcal{G}$ defined on a network G is *paradox-free* if for every subnetwork H of G, $L(H) \geq L(G)$. Paradox-free instances do not suffer from the Braess's paradox and their PoA cannot be improved by edge removal. An instance $\mathcal{G}$ is *paradox-ridden* if there is a subnetwork H of G such that $L(H) = L^*(G) = L(G)/\rho(G)$. Namely, the PoA of paradox-ridden instances can decrease to 1 by edge removal.

Best Subnetwork. Given instance $\mathcal{G}$, the *best subnetwork* H^B is a subnetwork of G that minimizes the equilibrium latency, i.e. H^B has $L(H^B) \leq L(H)$ for any subnetwork H of G.

Problem Definitions. We now introduce three basic problems regarding selfish network design:

- **Paradox-Ridden Recognition** (ParRid): Given an instance $\mathcal{G}$, decide if $\mathcal{G}$ is paradox-ridden.
- **Best Subnetwork Equilibrium Latency** (BSubEL) : Given an instance $\mathcal{G}$ defined on a network G, find the best subnetwork H^B of G and its equilibrium latency $L(H^B)$.
- **Minimum Latency Modification** (MinLatMod): Given an instance $\mathcal{G}$ defined on a network $G(V,E)$ with a polynomial latency $\ell_e(x) = \sum_{i=0}^{d} a_{e,i}x^i$, $a_{e,i} \geq 0$, for each $e \in E$, find modified latencies $\tilde{\ell}_e(x) = \sum_{i=0}^{d} \tilde{a}_{e,i}x^i$, $\tilde{a}_{e,i} \geq 0$, $e \in E_o$, so that the Euclidean distance of the vectors $(a_{e,i})_{e\in E_o, i\in[d]}$ and $(\tilde{a}_{e,i})_{e\in E_o, i\in[d]}$ is minimum, and for the instance $\tilde{\mathcal{G}}_o$ defined on the network $G_o(V,E_o)$ with latencies $\tilde{\ell}_e(x)$, o is a Nash flow with common latency $L^*(\mathcal{G})$.

3 Recognizing Paradox-Ridden Instances

In this section, we present a polynomial-time algorithm for ParRid on instances with strictly increasing linear latencies. We start with a lemma that reduces ParRid to the problem of checking if some optimal flow o is a Nash flow on G_o.

Lemma 1. *An instance $\mathcal{G}$ defined on a network $G(V,E)$ is paradox-ridden iff there is an optimal flow o that is a Nash flow on the subnetwork $G_o(V,E_o)$.*

For instances with strictly increasing linear latencies, the optimal flow is unique and can be efficiently computed. Then, checking whether the optimal flow o is a Nash flow on G_o can be performed by a shortest path computation.

Theorem 1. ParRid *can be decided in polynomial time for instances with strictly increasing linear latency functions.*

Proof. Computing the unique optimal flow o for an instance $\mathcal{G}$ with strictly increasing linear latencies can be performed in polynomial time (see e.g. [24]). To check whether o is a Nash flow on the subnetwork $G_o(V, E_o)$, we compute the length $d(v)$ of the shortest $s - v$ path wrt the edge lengths $\{\ell_e(o_e)\}_{e \in E_o}$ for all vertices $v \in V$. Then o is a Nash flow if for every edge $(u, v) \in E_o$, $d(v) = d(u) + \ell_{(u,v)}(o_{(u,v)})$ (see e.g. [29, Proposition 2.10]). □

Dealing with Constant Latencies. Next we formulate a general sufficient condition, under which ParRid can be decided in polynomial time for instances with arbitrary linear latencies. Let $\mathcal{G}$ be an instance defined on a network $G(V, E)$, let $E^c = \{e \in E : a_e = 0\}$ be the set of edges with constant latencies, let $E^i = E \setminus E^c$ be the set of edges with strictly increasing latencies, and let $\mathcal{O}$ be the set of different optimal flows of $\mathcal{G}$.

All optimal flows assign the same traffic to the edges with strictly increasing latencies, and can differ only on edges with constant latencies. Given a fixed optimal flow o, we formulate a Linear Program whose feasible solutions correspond to all ($\mathcal{G}$-feasible) flows that agree with the optimal flows on the edges in E^i :

$$\min \sum_{e \in E^c} f_e b_e\,, \qquad \text{s.t.}$$

$$\sum_{u:(v,u)\in E^i} o_{(v,u)} + \sum_{u:(v,u)\in E^c} f_{(v,u)} = \sum_{u:(u,v)\in E^i} o_{(u,v)} + \sum_{u:(u,v)\in E^c} f_{(u,v)} \qquad \forall v \in V \setminus \{s,t\}$$

$$\sum_{u:(s,u)\in E^i} o_{(s,u)} + \sum_{u:(s,u)\in E^c} f_{(s,u)} = 1 \qquad \text{(LP)}$$

$$\sum_{u:(u,t)\in E^i} o_{(u,t)} + \sum_{u:(u,t)\in E^c} f_{(u,t)} = 1$$

$$f_e \geq 0 \qquad \forall e \in E^c$$

(LP) has a variable f_e for each edge $e \in E^c$, while all o-related terms are fixed and determined by o. An optimal solution to (LP) corresponds to a $\mathcal{G}$-feasible flow that agrees with o on all edges in E^i and allocates traffic to the edges in E^c so that the total latency is minimized. Hence, every optimal solution to (LP) corresponds to an optimal flow. On the other hand, every optimal flow o' has $o_e = o'_e$ for all $e \in E^i$, and is translated into an optimal solution to (LP) by setting $f_e = o'_e$ for all $e \in E^c$. Therefore, there is a one-to-one correspondence between the optimal solutions to (LP) and the optimal flows in $\mathcal{O}$.

Given an optimal flow o, the discussion above reduces the problem of checking if there is a $o' \in \mathcal{O}$ that is a Nash flow on $G_{o'}$ to the problem of generating all optimal solutions of (LP) and checking whether some of them can be translated into a Nash flow on the corresponding subnetwork. This can be performed in polynomial time if (LP)'s optimal solution is unique (see e.g. [22, Theorem 2] on how to efficiently decide uniqueness of the optimal solution). Thus,

Theorem 2. ParRid *can be decided in polynomial time for instances with linear latency functions where (LP) has a unique optimal solution.*

In fact, it suffices to generate all optimal basic feasible solutions. This is true because (LP) allocates traffic to constant latency edges only. Hence, if a feasible solution f can be translated into a Nash flow on the corresponding subnetwork, this holds for any other feasible solution f' with $\{e : f'_e > 0\} \subseteq \{e : f_e > 0\}$. Therefore, the approach above can be extended to instances where (LP) has a small number of basic feasible solutions (i.e. polynomially many in m). This class includes instances with a constant number of constant latency edges.

4 Approximating the Best Subnetwork

Networks with Polynomially Many Short Paths. We present a subexponential-time approximation scheme for BSubEL on networks with polynomially many paths, each of polylogarithmic length. We first show that any flow (on any network) admits an ε-approximate "sparce" flow, which assigns traffic to $\mathrm{O}(\log m/\varepsilon^2)$ paths. The proof builds on the proof of Althöfer's Lemma [2].

Lemma 2. *Let $\mathcal{G}$ be an instance on a network $G(V,E)$, and let f be any $\mathcal{G}$-feasible flow. For any $\varepsilon > 0$, there exists a $\mathcal{G}$-feasible flow $\tilde{f}$ that assigns positive traffic to at most $\lfloor \log(2m)/(2\varepsilon^2) \rfloor + 1$ paths, such that $|\tilde{f}_e - f_e| \leq \varepsilon$, for $e \in E$.*

Proof. For convenience, we let $\mu = |\mathcal{P}|$ denote the number of paths in G, and index the $s-t$ paths in G by integers in $[\mu]$. Since the traffic rate is normalized to 1, we can interpret the flow f as a probability distribution on the set of paths $\mathcal{P}$. We prove that if we select $k > \log(2m)/(2\varepsilon^2)$ paths uniformly at random with replacement according to (the probability distribution) f, and assign to each path j a flow equal to the number of times j is selected divided by k, we obtain a flow that is an ε-approximation to f with positive probability. By the *Probabilistic Method* [1], such a flow exists.

Let ε be any fixed positive number, and let $k = \lfloor \log(2m)/(2\varepsilon^2) \rfloor + 1$. We define k independent identically distributed random variables $P_1, \dots, P_k$, each taking an integer value in $[\mu]$ according to distribution f. Namely, for all $i \in [k]$ and $j \in [\mu]$, $\mathbb{P}[P_i = j] = f_j$. For each path $j \in [\mu]$, let F_j be a random variable defined as $F_j = |\{i \in [k] : P_i = j\}|/k$. By linearity of expectation, $\mathbb{E}[F_j] = f_j$. For each edge e and each random variable P_i, we define an indicator variable $F_{e,i}$ that is 1 if e is included in the path indicated by P_i, and 0 otherwise. Since the random variables $\{P_i\}_{i \in [k]}$ are independent, for every fixed edge e, the variables $\{F_{e,i}\}_{i \in [k]}$ are independent as well. In addition, for every edge e, let $F_e = \frac{1}{k} \sum_{i=1}^{k} F_{e,i}$. We observe that $F_e = \sum_{j : e \in j} F_j$, and that $\mathbb{E}[F_e] = f_e$.

Since $\sum_{j=1}^{\mu} F_j = 1$, we can interpret the value of each F_j as an amount of flow assigned to path j, and the value of each F_e as an amount of flow assigned to edge e. Then the random variables $F_1, \dots, F_\mu$ define a ($\mathcal{G}$-feasible) flow on G that assigns positive traffic to at most k paths and agrees with f on expectation. By applying the Chernoff-Hoeffding bound [12], we obtain that for every edge e,

$$\mathbb{P}[|F_e - f_e| > \varepsilon] \leq 2\mathrm{e}^{-2\varepsilon^2 k} < 1/m\,,$$

where we use that $k > \log(2m)/(2\varepsilon^2)$. By applying the union bound, we obtain that $\mathbb{P}[\exists e : |F_e - f_e| > \varepsilon] < m(1/m) = 1$. Therefore, for any integer $k > \log(2m)/(2\varepsilon^2)$, there is positive probability that the ($\mathcal{G}$-feasible) flow $(F_1, \dots, F_\mu)$ satisfies $|F_e - f_e| \leq \varepsilon$ for all $e \in E$. By the Probabilistic Method, there exists a flow $\tilde{f}$ with the properties of $(F_1, \dots, F_\mu)$. □

For any $\varepsilon > 0$, let $\epsilon_1 > 0$ depend on ε and on some parameters of $\mathcal{G}$. By Lemma 2, there exists an ϵ_1-approximation $\tilde{f}$ to a Nash flow f on the best subnetwork $L(H^B)$ that assigns positive traffic to at most $\lfloor \log(2m)/(2\epsilon_1^2) \rfloor + 1$ paths. If G has polynomially many paths, $\tilde{f}$ can be found in subexponential time by exhaustive search. Next we show that if all paths in G are relatively short, $\tilde{f}$ is an ε-Nash flow on $G_{\tilde{f}}$, and all players' latencies in $\tilde{f}$ are at most $L(H^B) + \varepsilon/2$. Thus we obtain a subexponential approximation scheme for BSubEL.

Theorem 3. *Let $\mathcal{G} = (G(V, E), (a_e x + b_e)_{e \in E}, 1)$ be an instance with linear latencies, let $\alpha = \max_{e \in E}\{a_e\}$, and let H^B be the best subnetwork of G. For some constants $d_1, d_2 > 0$, let $|\mathcal{P}| \leq m^{d_1}$ and $|p| \leq \log^{d_2} m$, for all $p \in \mathcal{P}$. Then, for any $\varepsilon > 0$, we can compute in time $m^{O(d_1 \alpha^2 \log^{2d_2+1}(2m)/\varepsilon^2)}$ a flow $\tilde{f}$ that is an ε-Nash flow on $G_{\tilde{f}}$ and satisfies $\ell_p(\tilde{f}) \leq L(H^B) + \varepsilon/2$, for all paths p in $G_{\tilde{f}}$.*

Proof sketch. Let f be an acyclic Nash flow on the best subnetwork H^B, and for any $\varepsilon > 0$, let $\epsilon_1 = \varepsilon/(2\alpha \log^{d_2}(2m))$. Wlog. we assume that H^B is precisely G_f. By Lemma 2, there exists a $\mathcal{G}$-feasible acyclic flow $\tilde{f}$ on H^B that assigns positive flow to at most $k = \lfloor \log(2m)/(2\epsilon_1^2) \rfloor + 1 = \lfloor 2\alpha^2 \log^{2d_2+1}(2m)/\varepsilon^2 \rfloor + 1$ paths, and satisfies $|f_e - \tilde{f}_e| \leq \epsilon_1$ for all edges e in H^B, and $\tilde{f}_e = 0$ for all edges e not in H^B. Using that f is a Nash flow on H^B with $L(f) = L(H^B)$, we show that for any path p in the subnetwork $G_{\tilde{f}}$ determined by the edges used by $\tilde{f}$, $|\ell_p(f) - L(H^B)| \leq \varepsilon/2$. Therefore, there exists a $\mathcal{G}$-feasible flow $\tilde{f}$ that assigns positive flow to at most k paths, is an ε-Nash flow on $G_{\tilde{f}}$, and satisfies $|\ell_p(f) - L(H^B)| \leq \varepsilon/2$, for all paths p in $G_{\tilde{f}}$. A flow with the properties of $\tilde{f}$ can be computed in time $m^{O(d_1 k)}$ by exhaustive search. □

Instances with Strictly Increasing Latencies. For such instances, we show how to efficiently approximate the equilibrium latency on the best subnetwork within a factor less than the inapproximability threshold of 4/3.

Theorem 4. *For instances with strictly increasing linear latencies,* BSubEL *can be approximated in polynomial time within a factor of $4/3 - \delta$, where $\delta > 0$ depends on the instance.*

Proof sketch. Let $\mathcal{G}$ be an instance with strictly increasing linear latencies defined on $G(V, E)$, and let H^B be the best subnetwork of $\mathcal{G}$. If $\mathcal{G}$ is paradox-ridden, by Theorem 1, we can recognize it and compute H^B and $L(H^B)$ in polynomial time. Hence for paradox-ridden instances, we have an approximation ratio of 1.

If $\mathcal{G}$ is not paradox-ridden, we use the trivial algorithm that returns the entire network G. Since $\mathcal{G}$ is not paradox-ridden, $L(H^B) > L^*(G)$. Setting $\delta = \rho(G)(L(H^B) - L^*(G))/L(H^B) > 0$, we obtain that $L(G)/L(H^B) = \rho(G) - \delta$. Since G has linear latencies, $\rho(G) \leq 4/3$, and the theorem follows. □

5 Enforcing the Optimal Flow by Latency Modifications

Despite our positive results, there are instances where either finding the best subnetwork is hard, or the equilibrium latency on the best subnetwork is not close to the optimal average latency. For such instances, we present a polynomial-time algorithm that enforces the optimal flow by performing a minimal amount of latency modifications on the edges used by the optimal flow.

Theorem 5. MinLatMod *can be solved in polynomial time for instances with polynomial latency functions.*

Proof. Let $\mathcal{G}$ be an instance defined on a network $G(V, E)$ with a polynomial latency function $\ell_e(x) = \sum_{i=0}^{d} a_{e,i} x^i$, $a_{e,i} \geq 0$, for each $e \in E$. We can efficiently compute an optimal flow o within any specified accuracy (see e.g. [11]) and the corresponding subnetwork $G_o(V, E_o)$.

Let $\alpha = (a_{e,i})_{e \in E_o, i \in [d]}$ be coefficients vector of the latency functions for the edges used by the optimal flow o. We seek a modified coefficients vector $\tilde{\alpha} = (\tilde{a}_{e,i})_{e \in E_o, i \in [d]}$ so that the Euclidean distance of α and $\tilde{\alpha}$ is minimized, and for the instance $\tilde{\mathcal{G}}_o$ defined on G_o with latency functions $\tilde{\ell}_e(x) = \sum_{i=0}^{d} \tilde{a}_{e,i} x^i$, $\tilde{a}_{e,i} \geq 0$, $e \in E_o$, the flow o is a Nash flow with common latency $L^*(G)$. The best vector $\tilde{\alpha}$ is given by the optimal solution to the following Quadratic Program:

$$\begin{aligned}
&\min \sum_{e \in E_o} \sum_{i=1}^{d} (a_{e,i} - \tilde{a}_{e,i})^2 \\
\text{s.t.} \quad &\sum_{e \in p} \sum_{i=0}^{d} \tilde{a}_{e,i}\, o_e^i = L^*(G) \qquad \forall \text{paths } p \text{ in } G_o \qquad \text{(QP)} \\
&\tilde{a}_{e,i} \geq 0 \qquad \forall e \in E_o\,,\ \forall i \in [d]
\end{aligned}$$

The equality constraints ensure that all paths in G_o have a common latency $L^*(G)$ in o wrt the modified latency functions $\tilde{\ell}$. Thus o is a Nash flow with common latency $L^*(G)$ for the modified instance $\tilde{\mathcal{G}}_o$. (QP) always admits a feasible solution (see e.g. [25, Cor. 2.7]). Moreover, (QP) is a convex separable Quadratic Program, and can be solved in polynomial time within any specified accuracy (see e.g. [11]). □

Remark 1. We can use the same approach to compute a modified coefficients vector that turns the optimal flow o into a Nash flow on G_o wrt to the modified latencies with *any prescribed common latency Λ.*

References

1. Alon, N., Spencer, J.: The Probabilistic Method. John Wiley, Chichester (1992)
2. Althöfer, I.: On Sparse Approximations to Randomized Strategies and Convex Combinations. Linear Algebra and Applications 99, 339–355 (1994)

3. Azar, Y., Epstein, A.: The Hardness of Network Design for Unsplittable Flow with Selfish Users. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 41–54. Springer, Heidelberg (2006)
4. Bonifaci, V., Harks, T., Schäfer, G.: Stackelberg Routing in Arbitrary Networks. In: Papadimitriou, C., Zhang, S. (eds.) WINE 2008. LNCS, vol. 5385, pp. 239–250. Springer, Heidelberg (2008)
5. Braess, D.: Über ein Paradox aus der Verkehrsplanung. Unternehmensforschung 12, 258–268 (1968)
6. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Taxes for Linear Atomic Congestion Games. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 184–195. Springer, Heidelberg (2006)
7. Cole, R., Dodis, Y., Roughgarden, T.: How Much Can Taxes Help Selfish Routing? J. Comput. System Sci. 72(3), 444–467 (2006)
8. Fleischer, L., Jain, K., Mahdian, M.: Tolls for Heterogeneous Selfish Users in Multicommodity Networks and Generalized Congestion Games. In: Proc. of FOCS 2004, pp. 277–285 (2004)
9. Fotakis, D.: Stackelberg Strategies for Atomic Congestion Games. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 299–310. Springer, Heidelberg (2007)
10. Fotakis, D., Spirakis, P.: Cost-Balancing Tolls for Atomic Network Congestion Games. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 179–190. Springer, Heidelberg (2007)
11. Hochbaum, D.S., Shanthikumar, J.G.: Convex Separable Optimization is not Much Harder than Linear Optimization. J. ACM 37(4), 843–862 (1990)
12. Hoeffding, W.: Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
13. Kaporis, A.C., Spirakis, P.G.: The Price of Optimum in Stackelberg Games on Arbitrary Single Commodity Networks and Latency Functions. In: Proc. of SPAA 2006, pp. 19–28 (2006)
14. Karakostas, G., Kolliopoulos, S.: Edge Pricing of Multicommodity Networks for Heterogeneous Selfish Users. In: Proc. of FOCS 2004, pp. 268–276 (2004)
15. Karakostas, G., Kolliopoulos, S.: Stackelberg Strategies for Selfish Routing in General Multicommodity Networks. Algorithmica 53(1), 132–153 (2009)
16. Korilis, Y.A., Lazar, A.A., Orda, A.: Achieving Network Optima Using Stackelberg Routing Strategies. IEEE/ACM Trans. on Networking 5(1), 161–173 (1997)
17. Koutsoupias, E., Papadimitriou, C.: Worst-Case Equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
18. Lin, H., Roughgarden, T., Tardos, É.: A Stronger Bound on Braess's Paradox. In: Proc. of SODA 2004, pp. 340–341 (2004)
19. Lin, H., Roughgarden, T., Tardos, É., Walkover, A.: Braess's Paradox, Fibonacci Numbers, and Exponential Inapproximability. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 497–512. Springer, Heidelberg (2005)
20. Lipton, R.J., Markakis, E., Mehta, A.: Playing Large Games Using Simple Strategies. In: Proc. of EC 2003, pp. 36–41 (2003)
21. Lipton, R.J., Young, N.E.: Simple Strategies for Large Zero-Sum Games with Applications to Complexity Theory. In: Proc. of STOC 1994, pp. 734–740 (1994)
22. Mangasarian, O.L.: Uniqueness of Solution on Linear Programming. Linear Algebra and its Applications 25, 151–162 (1979)
23. Milchtaich, I.: Network Topology and the Efficiency of Equilibrium. Games and Economic Behavior 57, 321–346 (2006)

24. Minoux, M.: A Polynomial Algorithm for Minimum Quadratic Cost Flow Problems. European J. of Operational Research 18(3), 377–387 (1984)
25. Roughdarden, T., Tardos, É.: How Bad is Selfish Routing? J. ACM 49(2), 236–259 (2002)
26. Roughgarden, T.: The Price of Anarchy is Independent of the Network Topology. In: Proc. of STOC 2002, pp. 428–437 (2002)
27. Roughgarden, T.: Stackelberg Scheduling Strategies. SIAM J. on Computing 33(2), 332–350 (2004)
28. Roughgarden, T.: Selfish Routing and the Price of Anarchy. MIT Press, Cambridge (2005)
29. Roughgarden, T.: On the Severity of Braess's Paradox: Designing Networks for Selfish Users is Hard. J. Comput. System Sci. 72(5), 922–953 (2006)
30. Valiant, G., Roughgarden, T.: Braess's Paradox in Large Random Graphs. In: Proc. of EC 2006, pp. 296–305 (2006)