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.
Email: 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 that wants to improve equilibrium delays in selfish networks, is facing some basic questions:

- Is the network *paradox-ridden*?
- How can we delete some edges to *optimize* equilibrium flow delays?
- 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 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 on 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 Future and Emerging Technologies Unit of EC (IST priority – 6th FP), 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 Department of Information and Communication Systems Engineering, 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 are non-cooperative and selfish, and 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. [29, 32]).

Motivation and Previous Work. A Nash equilibrium may not optimize the network performance, usually 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*, which was suggested in a groundbreaking work by Koutsoupias and Papadimitriou [20]. The Price of Anarchy (PoA) is the ratio of the total latency of the Nash flow to the optimal total latency. Roughgarden [30] proved that the PoA for selfish routing 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, and $\frac{27+6\sqrt{3}}{23}$ for quadratic latencies). Moreover, Roughgarden presented a class of instances for which the bound of $\rho(\mathcal{D})$ is tight.

With the PoA for selfish routing 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* [19], where the administrator exploits a small fraction of centrally routed (aka 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 as the fraction of the coordinated traffic increases (see e.g. [31, 17, 4], and [11] for the case of atomic players with unsplittable traffic). Unfortunately, there are single-commodity instances for which the PoA remains unbounded under any allocation [4], and instances where enforcing the optimal flow may require a large fraction of the coordinated traffic [15]. A different approach is to introduce economic incentives, usually modeled as 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, a set of tolls that enforce the optimal flow can be computed efficiently even for heterogeneous players, who may have different latency-vs-tolls valuations (see e.g. [7, 10, 16], and [6, 12] for positive and negative results on refundable tolls for the case of atomic players with unsplittable traffic). 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. [10]).

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 illustrating example). Thus, given 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, more natural, and more appealing to both the network administrator and the players. From the administrator's point of view, blocking the traffic on some edges is arguably easier and less expensive to implement than setting up a mechanism for collecting tolls on every edge and refunding them to the players. From the players' point of view, edge removal is applied only if it results in a (significant) improvement on

their equilibrium latency, which is arguably preferable to either a toll mechanism, that increases the disutility of all players, or a Stackelberg strategy, that allocates the coordinated traffic to slower paths.

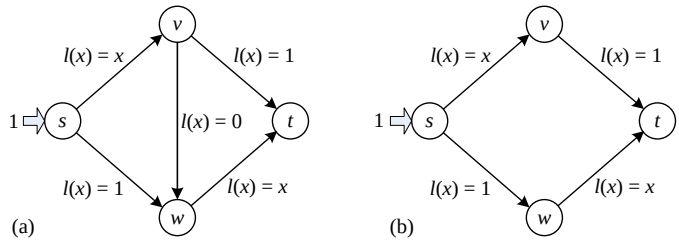

Fig. 1. (a). The optimal flow routes one half unit of traffic on the upper path s, v, t and one half unit on the lower path s, w, t, and achieves a total latency of $3/2$. In the (unique) Nash flow, all traffic goes through the path s, v, w, t. The players' latency is 2, which gives a PoA of $4/3$. (b). In the subnetwork obtained by removing the edge (v, w), the Nash flow coincides with the optimal flow. Hence the network on the left is *paradox-ridden*, and the network on the right is its *best subnetwork*.

Recent work indicates that edge removal can offer a tangible improvement on the performance of real-world networks (see e.g. [18, 32]). In this vein, Valiant and Roughgarden [34] proved that the Braess's paradox occurs with high probability on random networks, and that for a natural distribution of linear latencies, edge removal can improve the equilibrium latency by a factor arbitrarily close to $4/3$ (i.e. the worst-case PoA for linear latencies) with high probability (see also the references in [34] for other results in the same spirit).

Unfortunately, Roughgarden [33] 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 on the original network (i.e., edge removal can decrease the PoA to 1). Furthermore, Roughgarden proved 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 [33], we show that it becomes *polynomially solvable* for the important case of strictly increasing linear latencies[4]. Our starting point is the observation that 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 [33] 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

[4] We note that constant latency edges represent links of practically infinite capacity. Therefore real-world instances are most unlikely to contain a large number of constant latency edges, if they contain any.

strictly increasing linear latencies (cf. Theorem 1). Furthermore, we reduce the problem of recognizing a paradox-ridden network 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 edges with constant latency (cf. Lemma 2 and 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 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 algorithm's running time is exponential in $\mathrm{poly}(\log m)/\varepsilon^2$, where m denotes the number of edges (cf. Lemma 3 and Theorem 3). The analysis is based on a novel application of the Probabilistic Method (see e.g. [1]) motivated by Althöfer's "Sparsification" Lemma [2] and its application to the computation of approximate Nash equilibria for bimatrix games [25, 24]. To the best of our knowledge, this is the first time that similar techniques are applied in the context of selfish routing and congestion games.

Moreover, we show that for instances with strictly increasing linear latencies that are not paradox-ridden, there is an instance-dependent $\delta > 0$, such that the equilibrium latency (on the original network) 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[5] 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 of the remaining ones. In the last part of the paper, 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[6]. For the problem of finding the best subnetwork in the atomic model with unsplittable traffic, Azar and Epstein [3] obtained strong inapproximability results similar to those in [33]. In particular, they proved that for linear and polynomial latency functions, it is NP-hard to approximate the Nash equilibrium total latency on the best subnetwork within any factor smaller than the worst-case PoA for the corresponding class of games.

Interestingly, the Braess's paradox can be dramatically more severe in multi-commodity instances than in single-commodity ones. More precisely, Lin *et al.* [22] 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.* [23] 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 [27] 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.

[5] The reduction of [33, Theorem 3.3] constructs instances where almost all edges have constant latency 0. Using some very slowly increasing linear latency (namely, 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.

[6] Due to space limitations, we have restricted the discussion of related work to the most relevant results on the detection and elimination of the Braess's paradox. Nevertheless, there has been a large body of work on quantifying and mitigating the consequences of the Braess's paradox on selfish traffic, especially in the areas of Transportation Science and Computer Networks. The interested reader may e.g. see [33] for an extensive list of references.

2 Model, Preliminaries, Problem Definitions, and Results

Selfish Routing Instance. 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 $n = |V|$, let $m = |E|$, and let $\mathcal{P}$ (or $\mathcal{P}_G$, whenever the network G is not clear from the context) denote the (non-empty) 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 linear latency function is *constant* if $a_e = 0$.

Subnetworks and Subinstances. 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 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 e. Two 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 corresponding subnetwork of G. A flow f is acyclic if G_f contains no cycles.

Optimal Flow. The *total latency* of flow f, denoted $C(f)$, 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 an instance $\mathcal{G}$ defined on a network G and an optimal flow o, $G_o(V,E_o)$ is the subnetwork of G determined by the edges used by o.

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). The precise statements of these properties, along with other useful properties of optimal flows can be found in the Appendix.

Nash Flow. 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 equilibrium flow*, or simply 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 $C(f) = rL(f)$. We note that a Nash flow f on a network $G(V,E)$ is a Nash flow on any subnetwork $G'(V,E')$ of G with $E_f \subseteq E'$.

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. Lemma 9 in the Appendix). Hence, for a fixed 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 in polynomial time, while for strictly increasing latencies, the Nash flow is unique. The precise statements of these properties, along with other useful properties of Nash flows can be found in the Appendix.

A Characterization of Nash and Optimal Flows. For any instance $\mathcal{G} = (G(V,E),(\ell_e)_{e\in E},r)$ with convex latency functions, a ($\mathcal{G}$-feasible) flow f is a Nash flow iff for any ($\mathcal{G}$-feasible) flow g, $\sum_{e\in E} f_e\ell_e(f_e) \le \sum_{e\in E} g_e\ell_e(f_e)$ (see e.g. [8, (2.4)]). A flow o is optimal iff it is a Nash flow for the instance $\mathcal{G}' = (G(V,E),(\ell^*_e)_{e\in E},r)$, where $\ell^*_e(x) = d(x\ell_e(x))/dx$ (see e.g. [29, Corollary 2.7]). Using the previous characterization of Nash flows, we obtain that a ($\mathcal{G}$-feasible) flow o is optimal iff for any ($\mathcal{G}$-feasible) flow g,

$$\sum_{e\in E} o_e\ell^*_e(o_e) \le \sum_{e\in E} g_e\ell^*_e(o_e) \tag{1}$$

ε-Nash flow. 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', $\ell_p(f) \le \ell_{p'}(f) + \varepsilon$.

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, $\rho(\mathcal{G}) = L(\mathcal{G})/L^*(\mathcal{G})$. For linear latencies, $\rho(\mathcal{G}) \le 4/3$ [30], while for general latencies, $\rho(\mathcal{G}) \le \lfloor n/2 \rfloor$ [33].

Other Notation and Conventions. For any integer $k \ge 1$, 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[7]. 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 simply identify a selfish routing instance with the corresponding network. Thus, given an instance $\mathcal{G}$ defined on a network G, we write $L(G)$ (resp. $L^*(G)$) to denote the equilibrium (resp. optimal) latency of $\mathcal{G}$, and $\rho(G)$ to denote the PoA of $\mathcal{G}$.

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, $L(H) \ge L(G)$. Paradox-free instances do not suffer from the Braess's paradox and their PoA cannot be improved by edge removal. If the instance is not paradox-free, edge removal can decrease the equilibrium latency by a factor greater than 1 and at most $\rho(G)$. 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 minimizing the equilibrium latency, i.e. $L(H^B) \le L(H)$ for any subnetwork H of G.

2.1 Problem Definitions and Results

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 $\ell_e(x) = \sum_{i=0}^{d} a_{e,i}x^i$, $a_{e,i} \ge 0$, for each $e \in E$, find a set of modified latencies $\tilde{\ell}_e(x) = \sum_{i=0}^{d} \tilde{a}_{e,i}x^i$, $\tilde{a}_{e,i} \ge 0$, $e \in E_o$, 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 $L^*(\mathcal{G})$.

[7] Given an instance $\mathcal{G}$ with traffic rate $r > 0$, we can modify the (linear or polynomial) latency functions and construct an instance $\mathcal{G}'$ with traffic rate 1 so that (i) a flow is optimal (resp. Nash) for $\mathcal{G}$ iff it is optimal (resp. Nash) for $\mathcal{G}'$, (ii) $\rho(G) = \rho(G')$, and (iii) a best subnetwork of $\mathcal{G}$ is a best subnetwork of $\mathcal{G}'$ and vice versa.

Roughgarden [33] proved that ParRid is NP-hard to decide even for instances with latency functions in the class $\{x, 1, 0\}$, and BSubEL is NP-hard to approximate within a factor of $4/3 - \varepsilon$ for linear latencies, and within a factor of $\lfloor n/2 \rfloor - \varepsilon$ for general latencies, for any constant $\varepsilon > 0$. The only known approximation algorithm for BSubEL is the trivial one, which returns the entire network and achieves an approximation ratio of $4/3$ for linear latencies and $\lfloor n/2 \rfloor$ for general latencies.

Results. We obtain the first polynomial-time algorithms for ParRid, in case of strictly increasing linear latencies, and for $\mathrm{MinLatMod}$, and polynomial-time approximation algorithms for BSubEL that improve on the trivial algorithm for two interesting settings.

For ParRid, we provide a polynomial-time algorithm that decides it on instances with strictly linear increasing latencies (cf. Theorem 1). We extend our method to instances with linear latencies where constant latencies are allowed, under a general condition that restricts the number of different optimal flows (cf. Theorem 2). In particular, we show that ParRid reduces to generating all optimal basic feasible solutions of a Linear Program that describes the optimal traffic allocations to the constant latency edges. This task can be efficiently performed if the number of constant latency edges is small, which is usually the case in real-world networks.

For BSubEL, we first consider networks with polynomially many paths, each of polylogarithmic length, and linear latencies. In this setting, we provide a subexponential-time approximation scheme, which for any $\varepsilon > 0$, computes a subnetwork with an ε-Nash flow in which the players' latencies are at most $L(H^B) + \varepsilon/2$. The algorithm's running time is exponential in $\mathrm{poly}(\log m)/\varepsilon^2$ (cf. Lemma 3 and Theorem 3). Moreover, we show that for networks G that are not paradox-ridden and have strictly increasing linear latencies, there is an instance-dependent $\delta > 0$, such that $L(G) \leq (4/3 - \delta)L(H^B)$. Using the algorithm for ParRid to recognize paradox-ridden instances and the trivial algorithm for the remaining ones, we obtain a polynomial-time algorithm for BSubEL that achieves an approximation ratio smaller than $4/3$ for instances with strictly increasing linear latencies (cf. Theorem 4).

For $\mathrm{MinLatMod}$, we show how to reduce it to the solution of a convex quadratic separable minimization problem, thus obtaining a polynomial-time algorithm for it (cf. Theorem 5). This approach is quite useful when either finding the best subnetwork is hard, or the equilibrium latency on the best subnetwork is not close to the optimal average latency. In such cases, the algorithm for $\mathrm{MinLatMod}$ can enforce the optimal flow by keeping the network modifications (and the cost for implementing them) at a minimal level, while not increasing the players' disutility, as refundable tolls do.

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 the following lemma that reduces ParRid to the problem of checking whether there is some optimal flow o that is a Nash flow on G_o.

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

Proof. If there is an optimal flow o that is a Nash flow on G_o, then G_o is a subnetwork of G with $L(G_o) = L^*(G)$, and $\mathcal{G}$ is paradox-ridden. If $\mathcal{G}$ is paradox-ridden, let H be a subnetwork of G with $L(H) = L^*(G)$, and let f be a Nash flow on H. The flow f is a Nash flow on $G_f(V, E_f)$ as well. Moreover, since $L(f) = L^*(G)$, f is an optimal flow of $\mathcal{G}$. Hence, there is an optimal flow that is a Nash flow on the subnetwork determined by its used edges. □

For instances with strictly increasing linear latencies, the optimal flow is unique (see e.g. Remark 1 in the Appendix) 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. Hence we obtain that:

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 reduces to solving a convex quadratic separable min-cost flow problem, which can be performed in polynomial time (see e.g. Lemma 7 in the Appendix). 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. [33, Proposition 2.10]). Since the optimal flow o is unique, we use Lemma 1 and decide whether $\mathcal{G}$ is paradox-ridden in polynomial-time. □

Dealing with Constant Latencies. Next we formulate a general sufficient condition, under which ParRid can be decided in polynomial time for instances with (not necessarily increasing) linear latencies. Let $\mathcal{G}$ be an instance defined on a network $G(V, E)$ with linear latencies $\ell_e(x) = a_e x + b_e$, 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 $\mathcal{G}$. If $|E^c| > 1$, it may be that $|\mathcal{O}| > 1$, in which case Lemma 1 reduces ParRid to examining if some optimal flow $o \in \mathcal{O}$ is a Nash flow on G_o. This can be performed efficiently, if there is a procedure that generates all optimal flows in polynomial time.

By Lemma 9 and Lemma 10 in the Appendix, all optimal flows assign the same traffic to the edges with strictly increasing latencies, and can differ only on edges with constant latencies. Assuming 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 with strictly increasing latencies.

$$
\begin{aligned}
& \min \sum_{e \in E^c} f_e b_e \\
\text{s.t.} \quad & \sum_{u:(v,u) \in E^i} o_{(v,u)} + \sum_{u:(v,u) \in E^c} f_{(v,u)} = \sum_{u:(u,v) \in E^i} o_{(u,v)} + \sum_{u:(u,v) \in E^c} f_{(u,v)} && \forall v \in V \setminus \{s,t\} \\
& \sum_{u:(s,u) \in E^i} o_{(s,u)} + \sum_{u:(s,u) \in E^c} f_{(s,u)} = 1 && \text{(LP)} \\
& \sum_{u:(u,t) \in E^i} o_{(u,t)} + \sum_{u:(u,t) \in E^c} f_{(u,t)} = 1 \\
& f_e \geq 0 && \forall e \in E^c
\end{aligned}
$$

(LP) has a variable f_e for each edge $e \in E^c$, while all o-related terms are fixed and determined by o. A feasible solution to (LP) corresponds to a $\mathcal{G}$-feasible flow that agrees with o (and any other optimal flow) on all edges in E^i. An optimal solution to (LP) corresponds to a 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 (note that the total latency on the edges in E^i, namely the term $\sum_{e \in E^i}(a_e o_e^2 + b_e o_e)$, is fixed for all feasible solutions). 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 $\mathcal{O}$. Hence, we obtain the following:

Lemma 2. *The number of optimal flows of a selfish routing instance $\mathcal{G}$ with linear latencies is equal to the number of optimal solutions to (LP). Moreover, each optimal solution to (LP) can be translated into an optimal flow of $\mathcal{G}$, and vice versa.*

Given an optimal flow o, Lemma 2 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. [26, Theorem 2] on how to decide the uniqueness of the optimal solution of a Linear Program). 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 the optimal basic feasible solutions, since (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. polynomial many in m). For example, this class includes instances with a constant number of constant latency edges.

4 Approximating the Best Subnetwork

We give two types of approximation for the best subnetwork and its equilibrium latency on instances with linear latencies. We first consider networks with polynomially many paths, each of polylogarithmic length, and present a subexponential-time approximation scheme for BSubEL.

Networks with Polynomially Many Short Paths. We first show that any flow can be approximated by a "sparce" flow that assigns traffic to at most a logarithmic (in m) number of paths. The proof is along the lines of the proof of Althöfer's "Sparsification" Lemma [2].

Lemma 3. *Let $\mathcal{G}$ be an instance defined 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 all edges 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 $[\mu]$. We recall that the traffic rate is normalized to 1. Then, we can interpret the flow f as a probability distribution on the set of paths $\mathcal{P}$, where f_j is the probability that path j is selected. 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* (see e.g. [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 $P_1, \ldots, P_k$, 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 $e \in E$ and each random variable P_i, $i \in [k]$, 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 be a random variable defined as the average of the indicator variables $F_{e,i}$, i.e. $F_e = \frac{1}{k}\sum_{i=1}^{k} F_{e,i}$. By the definition of the random variables $\{F_j\}_{j\in[\mu]}$, F_e is equal to the sum of F_j's over all paths j that include e. Formally, $F_e = \sum_{j:e\in j} F_j = \frac{1}{k}\sum_{i=1}^{k} F_{e,i}$. By linearity of expectation, $\mathbb{E}[F_e] = f_e$ for all edges 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 $F_1, \ldots, F_\mu$ define a ($\mathcal{G}$-feasible) flow on G that assigns positive traffic to at most k paths and agrees with f on expectation. It suffices to show that the probability that there is an edge e with $|F_e - f_e| > \varepsilon$ is less than 1. By applying the Chernoff-Hoeffding bound[8], we obtain that for each fixed edge e,

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

The first inequality holds because $\mathbb{E}[F_e] = f_e$, and F_e is the average of k independent $0/1$ random variables. For the second inequality, 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 $(F_1, \ldots, F_\mu)$, which assigns positive flow to at most k paths, 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 $(F_1, \ldots, F_\mu)$. □

For any $\varepsilon > 0$, let $\epsilon_1 > 0$ depend on ε and on some parameters of the instance $\mathcal{G}$. Lemma 3 guarantees the existence of 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 the network G has polynomially many paths, such an $\tilde{f}$ can be found in subexponential time by exhaustive search. In the following, 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 the BSubEL.

Theorem 3. *Let $\mathcal{G} = (G(V, E), (a_e x + b_e)_{e \in E}, 1)$ be an instance with linear latency functions, 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, 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*

$$m^{\mathrm{O}(d_1 \alpha^2 \log^{2d_2+1}(2m)/\varepsilon^2)}$$

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. Let $\varepsilon > 0$ be any fixed constant, let $\epsilon_1 = \varepsilon/(2\alpha \log^{d_2}(2m))$, and let f be a Nash flow on the best subnetwork H^B. Then, $L(f) = L(H^B)$. In the following, we assume wlog. that f is acyclic, and that H^B is precisely $G_f(V, E_f)$. By applying Lemma 3 to H^B and f with approximation parameter ϵ_1, we obtain that there exists a $\mathcal{G}$-feasible flow $\tilde{f}$ on H^B that assigns positive flow to at most

$$k = \left\lfloor \frac{\log(2m)}{2\epsilon_1^2} \right\rfloor + 1 = \left\lfloor \frac{2\alpha^2 \log^{2d_2+1}(2m)}{\varepsilon^2} \right\rfloor + 1$$

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 outside H^B. Since f is acyclic, $\tilde{f}$ is acyclic too.

Next we show that $\tilde{f}$ is an ε-Nash flow on the subnetwork $G_{\tilde{f}}(V, E_{\tilde{f}})$ determined by the edges used by $\tilde{f}$. For every $p \in \mathcal{P}_{G_{\tilde{f}}} \subseteq \mathcal{P}_{H^B}$, $\ell_p(f) = \sum_{e \in p}(a_e f_e + b_e) = L(H^B)$, since f is a Nash flow on H^B. In addition, since $\tilde{f}_e \leq f_e + \epsilon_1$ for all edges e in H^B, and thus for all edges in $G_{\tilde{f}}$,

$$\begin{aligned} \ell_p(\tilde{f}) &\leq \textstyle\sum_{e \in p}(a_e(f_e + \epsilon_1) + b_e) \\ &\leq L(H^B) + |p|\alpha\,\epsilon_1 \\ &\leq L(H^B) + \varepsilon/2\,, \end{aligned}$$

[8] We use the following form of the Chernoff-Hoeffding bound (see e.g. [14]): Let $X_1, \ldots, X_k$ be random variables independently distributed in $[0, 1]$, and let $X = \frac{1}{k}\sum_{i=1}^{k} X_i$. Then, for all $\varepsilon > 0$, $\mathbb{P}[|X - \mathbb{E}[X]| > \varepsilon] \leq 2\mathrm{e}^{-2\varepsilon^2 k}$, where $\mathrm{e} = 2.71\ldots$ is the basis of natural logarithms.

where the last inequality follows from the choice of ϵ_1 and the polylogarithmic upper bound on the length of the paths in G. Similarly, using that $\tilde{f}_e \geq f_e - \epsilon_1$ for all edges e in H^B, we show that for every path $p \in \mathcal{P}_{G_{\tilde{f}}}$, $\ell_p(\tilde{f}) \geq L(H^B) - \varepsilon/2$. Therefore, there exists a $\mathcal{G}$-feasible acyclic flow $\tilde{f}$ that assigns positive flow to at most k paths, is an ε-Nash flow on $G_{\tilde{f}}$, and satisfies $|\ell_p(\tilde{f}) - L(H^B)| \leq \varepsilon/2$ for all paths p in $G_{\tilde{f}}$.

A flow with the properties of $\tilde{f}$ can be computed in time $m^{\mathrm{O}(d_1 k)}$ by exhaustive search. In particular, for each multiset with k paths from $\mathcal{P}$, we generate the corresponding flow g in the same way that in the proof of Lemma 3, the flow $(F_1, \ldots, F_\mu)$ is constructed from (the actual values of) $P_1, \ldots, P_k$. If the subnetwork G_g is acyclic, we check whether g is an ε-Nash flow on G_g. This can be performed by computing the minimum and the maximum latency paths in G_g, and checking whether their latencies differ by at most ε. Since G_g is a directed acyclic network, the minimum and the maximum latency paths can be computed in polynomial time. Among all acyclic flows that are ε-Nash flows on the corresponding subnetworks, we return the one that minimizes the latency on the maximum latency path. Since exhaustive search encounters $\tilde{f}$, this latency cannot exceed $L(H^B)+\varepsilon/2$. Hence we return a flow that is an ε-Nash flow on the corresponding subnetwork, and has all its path latencies bounded from above by $L(H^B)+\varepsilon/2$. Since $\mathcal{G}$ has at most m^{d_1} different paths, there are at most $m^{d_1 k}$ different multisets with k paths from $\mathcal{P}$, and the exhaustive search takes at most $m^{\mathrm{O}(d_1 k)}$ time. □

Instances with Strictly Increasing Latencies. Next we focus on instances with strictly increasing linear latencies, and show how to 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. Let $\mathcal{G}$ be an instance with strictly increasing linear latencies defined on a network $G(V, E)$, 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 the best subnetwork H^B and its equilibrium latency $L(H^B)$ 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. Next we prove that there is an instance-dependent $\delta > 0$, such that $L(G)/L(H^B) \leq \rho(G) - \delta$. Since G has linear latencies, $\rho(G) \leq 4/3$, and the theorem follows.

Let f be the Nash flow on the best subnetwork H^B, and let o be the optimal flow of $\mathcal{G}$. Since $\mathcal{G}$ is not paradox-ridden, the flows f and o are different. By Taylor expansion for quadratic functions,

$$\sum_{e \in E}(a_e f_e^2 + b_e f_e) = \sum_{e \in E}(a_e o_e^2 + b_e o_e) + \sum_{e \in E}(2a_e o_e + b_e)(f_e - o_e) + \sum_{e \in E} a_e (f_e - o_e)^2 \quad \Leftrightarrow$$

$$C(f) = C(o) + \underbrace{\sum_{e \in E}(2a_e o_e + b_e)(f_e - o_e) + \sum_{e \in E} a_e (f_e - o_e)^2}_{= \sigma} \tag{2}$$

We note that $\sigma > 0$ because $\sum_{e \in E} a_e (f_e - o_e)^2 > 0$, since $a_e > 0$ for all $e \in E$ and f and o are different, and $\sum_{e \in E}(2a_e o_e + b_e)(f_e - o_e) \geq 0$, by the characterization of optimal flows by (1).

Using that the traffic rate is 1, we obtain that $C(f) = L(f) = L(H^B)$, and $C(o) = L^*(G)$. Moreover, by the definition of the PoA, $L^*(G) = L(G)/\rho(G)$. Therefore, (2) implies that

$$L(H^B) = L(G)/\rho(G) + \sigma \Rightarrow L(G)/L(H^B) = \rho(G) - \rho(G)\sigma/L(H^B)$$

Setting $\delta = \rho(G)\sigma/L(H^B) > 0$ concludes the proof of the theorem. □

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. Lemma 8 in the Appendix) 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, due to monetary reasons, to turn the optimal flow o into a Nash flow of average latency $L^*(G)$ on $G_o(V, E_o)$ by modifying α as little as possible. In particular, 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 coefficients vector $\tilde{\alpha}$ is given by the optimal solution to the following Quadratic Program:

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

The equality constraints of (QP) 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) is a convex separable Quadratic Program, and can be solved in polynomial time within any specified accuracy (see e.g. [21]). Moreover, (QP) always admits a feasible solution since the optimal flow o is a Nash flow on G_o with latency functions $\ell_e^*(x) = d(x\ell_e(x))/dx = \sum_{i=0}^{d}(i+1)a_{e,i}x^i$. Therefore, there is a $\Lambda > 0$, such that all paths in G_o have a common latency Λ wrt to the latency functions ℓ^*. Scaling the coefficients $\{(i+1)a_{e,i}\}_{e\in E_o, i\in[d]}$ uniformly by $L^*(G)/\Lambda$ gives a feasible solution to (QP). □

Remark. 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. In particular, the proof of Theorem 5 implies that by changing the rhs of the equality constraints in (QP) to Λ (instead of $L^*(G)$), we can efficiently compute a coefficients vector $\tilde{\alpha}$ so that the Euclidean distance of α and $\tilde{\alpha}$ is minimized, and the optimal flow o is a Nash flow with common latency Λ for the corresponding instance $\tilde{\mathcal{G}}_o$.

References

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

3. Y. Azar and A. Epstein. The hardness of network design for unsplittable flow with selfish users. In *Proc. of the 3rd Workshop on Approximation and Online Algorithms (WAOA '05)*, pp. 41–54, 2005.
4. V. Bonifaci, T. Harks, and G. Schäfer. Stackelberg Routing in Arbitrary Networks. In *Proc. of the 4th Workshop on Internet and Network Economics (WINE '08)*, pp. 239–250, 2008.
5. D. Braess. Über ein paradox aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.
6. I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Taxes for Linear Atomic Congestion Games. In *Proc. of the 14th European Symposium on Algorithms (ESA '06)*, pp. 184–195, 2006.
7. R. Cole, Y. Dodis, and T. Roughgarden. How Much Can Taxes Help Selfish Routing? *Journal of Computer and System Sciences*, 72(3):444–467, 2006.
8. J.R. Correa, A.S. Schulz, and N.E. Stier Moses. Selfish Routing in Capacitated Networks. *Mathematics of Operations Research*, 29(4):961–976, 2004.
9. J. Edmonds and R.M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. of the ACM*, 19(2):248–264, 1972.
10. L. Fleischer, K. Jain, and M. Mahdian. Tolls for Heterogeneous Selfish Users in Multicommodity Networks and Generalized Congestion Games. In *Proc. of the 45th IEEE Symp. on Foundations of Computer Science (FOCS '04)*, pp. 277–285, 2004.
11. D. Fotakis. Stackelberg strategies for atomic congestion games. In *Proc. of the 15th European Symposium on Algorithms (ESA '07)*, pp. 299–310, 2007.
12. D. Fotakis and P. Spirakis. Cost-Balancing Tolls for Atomic Network Congestion Games. In *Proc. of the 3rd Workshop on Internet and Network Economics (WINE '07)*, pp. 179–190, 2007.
13. D.S. Hochbaum and J.G. Shanthikumar. Convex separable optimization is not much harder than linear optimization. *J. of the ACM*, 37(4):843–862, 1990.
14. W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
15. A.C. Kaporis and P.G. Spirakis. The Price of Optimum in Stackelberg Games on Arbitrary Single Commodity Networks and Latency Functions. In *Proc. 18th ACM Symp. on Parallel Algorithms and Architect. (SPAA '06)*, pp. 19–28, 2006.
16. G. Karakostas and S. Kolliopoulos. Edge Pricing of Multicommodity Networks for Heterogeneous Selfish Users. In *Proc. of the 45th IEEE Symp. on Foundations of Computer Science (FOCS '04)*, pp. 268–276, 2004.
17. G. Karakostas and S. Kolliopoulos. Stackelberg Strategies for Selfish Routing in General Multicommodity Networks. *Algorithmica*, 53(1):132–153, 2009.
18. F. Kelly. *The Mathematics of Traffic in Networks*. In *The Princeton Companion to Mathematics* (Editors: T. Gowers, J. Green and I. Leader). Princeton University Press, 2008.
19. Y.A. Korilis, A.A. Lazar, and A. Orda. Achieving Network Optima Using Stackelberg Routing Strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.
20. E. Koutsoupias and C. Papadimitriou. Worst-Case Equilibria. In *Proc. of the 16thAnnual Symposium on Theoretical Aspects of Computer Science (STACS '99)*, pp. 404–413, 1999.
21. M.K. Kozlov, S.P. Tarasov, and L.G. Khachian. Polynomial Solvability of Convex Quadratic Programming. *Soviet Mathematics.*, Doklady 20:1108–1111, 1979.
22. H. Lin, T. Roughgarden, and É. Tardos. A Stronger Bound on Braess's Paradox. In *Proc. of the 15th ACM-SIAM Symp. on Discrete Algorithms (SODA '04)*, pp. 340–341, 2004.
23. H. Lin, T. Roughgarden, É. Tardos, and A. Walkover. Braess's Paradox, Fibonacci Numbers, and Exponential Inapproximability. In *Intl. Colloquium on Automata, Languages and Programming (ICALP '05)*, pp. 497–512, 2005.
24. R.J. Lipton, E. Markakis, and A. Mehta. Playing Large Games Using Simple Strategies. In *Proc. of the 4th ACM Conference on Electronic Commerce (EC '03)*, pp. 36–41, 2003.
25. R.J. Lipton and N.E. Young. Simple Strategies for Large Zero-Sum Games with Applications to Complexity Theory. In *Proc. of the 26th ACM Symp. on Theory of Computing (STOC '94)*, pp. 734–740, 1994.
26. O.L. Mangasarian. Uniqueness of Solution on Linear Programming. *Linear Algebra and Applicat.*, 25:151–162, 1979.
27. I. Milchtaich. Network Topology and the Efficiency of Equilibrium. *Games and Economic Behavior*, 57:321346, 2006.
28. M. Minoux. A Polynomial Algorithm for Minimum Quadratic Cost Flow Problems. *European Journal of Operational Research*, 18(3):377–387, 1984.
29. T. Roughdarden and É. Tardos. How Bad is Selfish Routing? *J. of the ACM*, 49(2):236–259, 2002.
30. T. Roughgarden. The Price of Anarchy is Independent of the Network Topology. In *Proc. of the 34th ACM Symp. on Theory of Computing (STOC '02)*, pp. 428–437, 2002.
31. T. Roughgarden. Stackelberg Scheduling Strategies. *SIAM Journal on Computing*, 33(2):332–350, 2004.
32. T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.
33. T. Roughgarden. On the Severity of Braess's Paradox: Designing Networks for Selfish Users is Hard. *Journal of Computer and System Sciences*, 72(5):922–953, 2006.
34. G. Valiant and T. Roughgarden. Braess's Paradox in Large Random Graphs. In *Proc. of the 7th ACM Conference on Electronic Commerce (EC '06)*, pp. 296–305, 2006.

A Basic Properties of Nash and Optimal Flows

Objective and Feasible Set. Nash flows and optimal flows are closely related, as it is suggested by the objective function for computing them via the following *Non-Linear Program* :

$$\text{(NLP)} \qquad \min_{f} \sum_{e \in E} \eta_e(f_e)\,, \text{ where } \eta_e(f_e) = \begin{cases} \ell_e(f_e) f_e & \text{for an optimal flow} \\ \displaystyle\int_0^{f_e} \ell_e(t)dt & \text{for a Nash flow} \end{cases}$$

For an instance $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$, the feasible set of (NLP) consists of all $\mathcal{G}$-feasible flows. For the instances considered in this paper, where the latency functions $\ell_e(x)$ are non-negative, non-decreasing, continuous, continuously differentiable, and convex, and there always exist an acyclic optimal flow and an acyclic Nash flow, there is no essential difference between letting the feasible set of (NLP) consist of all $\mathcal{G}$-feasible *edge* flows $(f_e)_{e\in E}$, and letting the feasible set of (NLP) consist of all $\mathcal{G}$-feasible *path* flows $(f_p)_{p\in\mathcal{P}}$. In the following, we call a latency function *standard* if it is non-negative, non-decreasing, continuous, continuously differentiable, and convex.

We observe that (NLP) is a *separable* program, i.e. each non-linear term is a function of a single variable. Furthermore, for standard latency functions, (NLP) is a *convex* program.

Interplay of Nash and Optimal Flows. An important consequence of the convexity of NLP is the following characterizations of optimal and Nash flows (see e.g. [32]):

Lemma 4. *Let $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$ be an instance with standard latency functions. A $\mathcal{G}$-feasible flow o is optimal for $\mathcal{G}$ iff it is a Nash flow for the instance $\mathcal{G}' = (G(V,E), (\ell^*_e)_{e\in E}, r)$, where $\ell^*_e(x) = d(x\ell_e(x))/dx$.*

Lemma 5. *Let $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$ be an instance with standard latency functions. A $\mathcal{G}$-feasible flow f is a Nash flow iff for any $\mathcal{G}$-feasible flow g,*

$$\sum_{e\in E} f_e \ell_e(f_e) \le \sum_{e\in E} g_e \ell_e(f_e)$$

Lemma 6. *Let $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$ be an instance with standard latency functions, and let $\ell^*_e(x) = d(x\ell_e(x))/dx$. A $\mathcal{G}$-feasible flow o is optimal iff for any $\mathcal{G}$-feasible flow g,*

$$\sum_{e\in E} o_e \ell^*_e(o_e) \le \sum_{e\in E} g_e \ell^*_e(o_e)$$

Time Complexity of (NLP). Minoux [28] proved that the scaling technique of Edmonds and Karp [9], which for linear costs gives a polynomial running time for the out-of-kilter method, can be applied to separable quadratic min-cost flow problems. Thus Minoux presented a polynomial-time algorithm for the quadratic min-cost flow problem.

Lemma 7. *(NLP) can be solved in polynomial time for instances with linear latency functions $\ell_e(x) = a_e x + b_e$ with non-negative rational coefficients a_e, b_e .*

Subsequently, Hochbaum and Shanthikumar [13] proved that solving convex separable min-cost flow problems is not much harder than linear optimization. More precisely, they presented an algorithm that finds a feasible solution whose components are within an additive term of ε from the optimal solution (this is called an ε-accurate solution). The algorithm's running time is polynomial in $\log(1/\varepsilon)$ and the input size. In simple words, the algorithm of Hochbaum and Shanthikumar computes the optimal solution to any specified accuracy in polynomial time.

Lemma 8. *For instances with standard latency functions, an ε-accurate solution to (NLP) can be computed in time polynomial in $\log(1/\varepsilon)$ and the input size.*

Existence and Uniqueness of Nash and Optimal Flows. The following lemma establishes that for instances with standard latency functions, a Nash flow exists and all Nash flows have the same common players' latency and total latency (see e.g. [32]).

Lemma 9. *Let $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$ be an instance with standard latency functions. Then, $\mathcal{G}$ admits a Nash flow f. Furthermore, if f, f' are Nash flows, then $\ell_e(f_e) = \ell_e(f'_e)$ for all $e \in E$, $\ell_p(f) = \ell_p(f')$ for all $p \in \mathcal{P}$, $L(f) = L(f')$, and $C(f) = C(f')$.*

Moreover, if the latency functions ℓ_e are strictly increasing, then the Nash flow is unique (see e.g. [32, Corollary 2.6.4]). We note that Lemma 10 does not rule out the possibility that some instances with constant latency functions also admit a unique Nash flow (see e.g. the instance in Fig. 1).

Lemma 10. *Let $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$ be an instance with standard strictly increasing latency functions. If f, f' are Nash flows, then $f_e = f'_e$ for all $e \in E$.*

Remark 1. Lemma 9 and Lemma 10 essentially follow from the convexity of (NLP). By similar arguments, one can prove that for an instance $\mathcal{G} = (G(V,E), (\ell_e)_{e\in E}, r)$ with standard latency functions, if o, o' are optimal flows, then $\ell_e(o_e) = \ell_e(o'_e)$, for all $e \in E$. In addition, if the latency functions are strictly increasing, then $o_e = o'_e$ for all $e \in E$.