On the Hardness of Network Design for Bottleneck Routing Games*

Dimitris Fotakis[1], Alexis C. Kaporis[2], Thanasis Lianeas[1], and Paul G. Spirakis[3,4]

[1] School of Electrical and Computer Engineering
National Technical University of Athens, 15780 Athens, Greece
[2] Department of Information and Communication Systems Engineering
University of the Aegean, 83200 Samos, Greece
[3] Department of Computer Engineering and Informatics
University of Patras, 26500 Patras, Greece
[4] Computer Technology Institute and Press - Diophantus
N. Kazantzaki Str., University Campus, 26500 Patras, Greece
fotakis@cs.ntua.gr, kaporisa@gmail.com,
tlianeas@mail.ntua.gr, spirakis@cti.gr

Abstract. In routing games, the network performance at equilibrium can be significantly improved if we remove some edges from the network. This counterintuitive fact, a.k.a. Braess's paradox, gives rise to the network design problem, where we seek to recognize routing games suffering from the paradox, and to improve the equilibrium performance by edge removal. In this work, we investigate the computational complexity and the approximability of network design for non-atomic bottleneck routing games, where the individual cost of each player is the bottleneck cost of her path, and the social cost is the bottleneck cost of the network. We first show that bottleneck routing games do not suffer from Braess's paradox either if the network is series-parallel, or if we consider only subpath-optimal Nash flows. On the negative side, we prove that even for games with strictly increasing linear latencies, it is NP-hard not only to recognize instances suffering from the paradox, but also to distinguish between instances for which the Price of Anarchy (PoA) can decrease to 1 and instances for which the PoA is $\Omega(n^{0.121})$ and cannot improve by edge removal. Thus, the network design problem for such games is NP-hard to approximate within a factor of $O(n^{0.121-\varepsilon})$, for any constant $\varepsilon > 0$. On the positive side, we show how to compute an almost optimal subnetwork w.r.t. the bottleneck cost of its worst Nash flow, when the worst Nash flow in the best subnetwork routes a non-negligible amount of flow on all edges. The running time is determined by the total number of paths, and is quasipolynomial if the number of paths is quasipolynomial.

* This work was supported by the project Algorithmic Game Theory, co-financed by the European Union (European Social Fund - ESF) and Greek national funds, through the Operational Program "Education and Lifelong Learning", under the research funding program Thales, by an NTUA Basic Research Grant (PEBE 2009), by the ERC project RIMACO, and by the EU FP7/2007-13 (DG INFSO G4-ICT for Transport) under Grant Agreement no. 288094 (Project eCompass).

M. Serna (Ed.): SAGT 2012, LNCS 7615, pp. 156–167, 2012.

1 Introduction

An typical instance of a non-atomic *bottleneck routing game* consists of a directed network, with origin s and destination t, where each edge has a non-decreasing function determining the edge's latency as a function of traffic. A traffic rate is controlled by an infinite population of players, each willing to route a negligible amount of traffic through an $s-t$ path. The players seek to minimize the maximum edge latency, a.k.a. the *bottleneck cost* of their path. Thus, the players reach a *Nash equilibrium flow*, or simply a *Nash flow*, where they all use paths with a common locally minimum bottleneck cost. Bottleneck routing games and their variants have received considerable attention due to their practical applications to communication networks (see e.g., [6,3] and the references therein).

Previous Work. Bottleneck routing games admit a Nash flow that is optimal for the network, in the sense that it minimizes the maximum latency on any used edge, a.k.a. the bottleneck cost of the network (see e.g., [3, Corollary 2]). However, bottleneck routing games usually admit many other Nash flows, some with a bottleneck cost quite far from the optimum. Hence, there has been a considerable interest in quantifying the performance degradation, due to the players' selfish behavior, in (several variants of) bottleneck routing games. This is measured by the *Price of Anarchy* (PoA) [13], that is the ratio of the bottleneck cost of the worst Nash flow to the optimal bottleneck cost of the network.

Simple examples (see e.g., [7, Fig. 2]) demonstrate that the PoA of bottleneck routing games with linear latencies can be $\Omega(n)$, where n is the number of nodes. For atomic splittable bottleneck routing games, where the population of players is finite, and each player has a non-negligible amount of traffic which can be split among different paths, Banner and Orda [3] observed that the PoA can be unbounded, even for very simple networks, if the players have different origins and destinations and the latency functions are exponential. On the other hand, Banner and Orda proved that if the players use paths that, as a secondary objective, minimize the number of bottleneck edges, then all Nash flows are optimal. For a variant of non-atomic bottleneck routing games, where the social cost is the average (instead of the maximum) bottleneck cost of the players, Cole, Dodis, and Roughgarden [7] proved that the PoA is 4/3, if the latency functions are affine and a subclass of Nash flows, called *subpath-optimal Nash flows*, is only considered. Subsequently, Mazalov et al. [16] studied the inefficiency of the best Nash flow under this notion of social cost.

For atomic unsplittable bottleneck routing games, where each player routes a unit of traffic through a single $s-t$ path, Banner and Orda [3] proved that for polynomial latencies of degree d, the PoA is $O(m^d)$, where m is the number of edges. On the other hand, Epstein, Feldman, and Mansour [8] proved that for series-parallel networks, all Nash flows are optimal. Busch and Magdon-Ismail [5] proved that the PoA of atomic unsplittable bottleneck routing games with identity latency functions can be bounded in terms of natural topological properties of the network. In particular, they proved that the PoA of such games is $O(l+\log n)$, where l is the length of the longest $s-t$ path, and $O(k^2+\log^2 n)$, where k is length of the longest circuit.

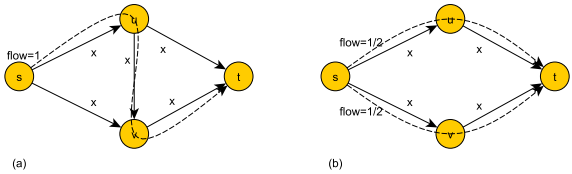

Fig. 1. Braess's paradox for bottleneck routing games. We consider identity latency functions and a unit of traffic to be routed from s to t. The worst Nash flow, in (a), has a bottleneck cost of 1. The optimal flow is the same as the flow in (b), and achieves a bottleneck cost of 1/2. Hence, PoA = 2. In the subnetwork (b), the Nash flow is unique and coincides with the optimal flow. Thus the PoA improves to 1. Hence the network on the left is *paradox-ridden*, and the network on the right is the *best subnetwork* of it.

With the PoA of bottleneck routing games so large and crucially depending on topological properties of the network, a natural approach to improving the equilibrium performance is to exploit Braess's paradox [4], namely that removing some edges may change the network topology (e.g., it may decrease the length of the longest path or cycle), and significantly improve the bottleneck cost of the worst Nash flow (see e.g., Fig. 1). This approach gives rise to the (selfish) *network design problem*, where we seek to recognize bottleneck routing games suffering from the paradox, and to improve the bottleneck cost of the worst Nash flow by edge removal. In particular, given a bottleneck routing game, we seek for the *best subnetwork*, namely, the subnetwork for which the bottleneck cost of the worst Nash flow is best possible. In this setting, one may distinguish two extreme cases: *paradox-free* instances, where edge removal cannot improve the bottleneck cost of the worst Nash flow, and *paradox-ridden* instances, where the bottleneck cost of the worst Nash flow in the best subnetwork is equal to the optimal bottleneck cost of the original network (see also [18,11]).

The approximability of selective network design, a generalization of network design where we cannot remove certain edges, was considered by Hou and Zhang [12]. For atomic unsplittable bottleneck routing games with a different traffic rate and a different origin and destination for each player, they proved that if the latency functions are polynomials of degree d, it is NP-hard to approximate selective network design within a factor of $O(m^{d-\varepsilon})$, for any $\varepsilon > 0$. Moreover, for atomic k-splittable bottleneck routing games with multiple origin-destination pairs, they proved that selective network design is NP-hard to approximate within any constant factor.

However, a careful look at the reduction of [12] reveals that their strong inapproximability results crucially depend on both (i) that we can only remove certain edges from the network, so that the subnetwork actually causing a large PoA cannot be destroyed, and (ii) that the players have different origins and destinations (and also are atomic and have different traffic rates). As for the importance of (ii), in a different setting, where the players' individual cost is the sum of edge latencies on their path and the social cost is the bottleneck cost of the network, it is known that Braess's paradox can be dramatically more

severe for instances with multiple origin-destination pairs than for instances with a single origin-destination pair. More precisely, Lin et al. [14] proved that if the players have a common origin and destination, the removal of at most k edges from the network cannot improve the equilibrium bottleneck cost by a factor greater than $k+1$. On the other hand, Lin et al. [15] presented an instance with two origin-destination pairs where the removal of a single edge improves the equilibrium bottleneck cost by a factor of $2^{\Omega(n)}$. Therefore, both at the technical and at the conceptual level, the inapproximability results of [12] do not really shed light on the approximability of the (simple, non-selective) network design problem in the simplest, and most interesting, setting of non-atomic bottleneck routing games with a common origin and destination for all players.

Contribution. In this work, we investigate the approximability of the network design problem for the simplest, and seemingly easier to approximate, variant of non-atomic bottleneck routing games with a single origin-destination pair. Our main result is that network design is hard to approximate within reasonable factors, and holds even for strictly increasing linear latencies. To the best of our knowledge, this is the first work that investigates the approximability of the network design problem for the basic variant of bottleneck routing games.

In Section 3, we use techniques similar to those in [8,7], and show that bottleneck routing games do not suffer from Braess's paradox either if the network is series-parallel, or if we consider only subpath-optimal Nash flows.

On the negative side, we employ, in Section 4, a reduction from the 2-Directed Disjoint Paths problem, and show that for linear bottleneck routing games, it is NP-hard to recognize paradox-ridden instances (Lemma 1). In fact, the reduction shows that it is NP-hard to distinguish between paradox-ridden instances and paradox-free instances, even if their PoA is equal to 4/3, and thus, it is NP-hard to approximate the network design problem within a factor less than 4/3.

In Section 5, we apply essentially the same reduction, but in a recursive way, and obtain a much stronger inapproximability result. We assume the existence of a γ-gap instance, which establishes that network design is inapproximable within a factor less than γ, and show that the construction of Lemma 1, but with some edges replaced by copies of the gap instance, amplifies the inapproximability threshold by a factor of 4/3, while it increases the size of the network by roughly a factor of 8 (Lemma 2). Therefore, starting from the 4/3-gap instance of Lemma 1, and recursively applying this construction a logarithmic number times, we show that it is NP-hard to approximate the network design problem for linear bottleneck routing games within a factor of $O(n^{0.121-\varepsilon})$, for any constant $\varepsilon > 0$. An interesting technical point is that we manage to show this inapproximability result, even though we do not know how to efficiently compute the worst equilibrium bottleneck cost of a given subnetwork. Hence, our reduction uses a certain subnetwork structure to identify good approximations to the best subnetwork. To the best of our knowledge, this is the first rime that a similar recursive construction is used to amplify the inapproximability threshold of the network design problem, and of any other optimization problem related to selfish routing.

In Section 6, we consider general latency functions, and present an algorithm for finding a subnetwork that is almost optimal w.r.t. the bottleneck cost of its worst Nash flow, when the worst Nash flow in the best subnetwork routes a non-negligible amount of flow on all edges. The algorithm is based on Althöfer's Sparcification Lemma [1], and is motivated by its recent application to network design for additive routing games [11]. For any constant $\varepsilon > 0$, the algorithm computes a subnetwork and an $\varepsilon/2$-Nash flow whose bottleneck cost is within an additive term of $O(\varepsilon)$ from the worst equilibrium bottleneck cost in the best subnetwork. The running time is roughly $|\mathcal{P}|^{\mathrm{poly}(\log m)/\varepsilon^2}$, and is quasipolynomial, when the number $|\mathcal{P}|$ of paths is quasipolynomial.

Next, we present our results with as much technical justification as the space constraints permit. The interested reader may find the omitted proofs in [10].

Other Related Work. Considerable attention has been paid to the approximability of network design for *additive routing games*, where the players seek to minimize the sum of edge latencies on their path, and the social cost is the total latency incurred by the players. Roughgarden [18] introduced the selfish network design problem in this setting, and proved that it is NP-hard to recognize paradox-ridden instances. He also proved that it is NP-hard to approximate the network design problem for such games within a factor less than 4/3 for affine latencies, and less than $\lfloor n/2 \rfloor$ for general latencies. For atomic unsplittable additive routing games with weighted players, Azar and Epstein [2] proved that network design is NP-hard to approximate within a factor less than 2.618, for affine latencies, and less than $d^{\Theta(d)}$, for polynomial latencies of degree d.

On the positive side, Milchtaich [17] proved that non-atomic additive routing games on series-parallel networks do not suffer from Braess's paradox. Fotakis, Kaporis, and Spirakis [11] proved that we can efficiently recognize paradox-ridden instances when the latency functions are affine, and all, but possibly a constant number of them, are strictly increasing. Moreover, applying Althöfer's Sparsification Lemma [1], they gave an algorithm that approximates network design for affine additive routing games within an additive term of ε, for any constant $\varepsilon > 0$, in time that is subexponential if the total number of $s-t$ paths is polynomial and all paths are of polylogarithmic length.

2 Model, Definitions, and Preliminaries

Routing Instances. A *routing instance* is a tuple $\mathcal{G} = (G(V,E), (c_e)_{e\in E}, r)$, where $G(V,E)$ is a directed network with origin s and destination t, $c_e : [0,r] \mapsto \mathbb{R}_{\geq 0}$ is a continuous non-decreasing latency function associated with edge e, and $r > 0$ is the traffic rate entering at s and leaving at t. We consider a non-atomic model of selfish routing, where r is divided among an infinite population of players, each routing a negligible amount of traffic from s to t. We let $n \equiv |V|$ and $m \equiv |E|$, and let $\mathcal{P}$ denote the set of simple $s-t$ paths in G. A latency function $c_e(x)$ is *linear* if $c_e(x) = a_e x$, for some $a_e > 0$, and *affine* if $c_e(x) = a_e x + b_e$, for some $a_e, b_e \geq 0$. We say that a latency function $c_e(x)$ satisfies the *Lipschitz condition* with constant $\xi > 0$, if for all $x, y \in [0,r]$, $|c_e(x) - c_e(y)| \leq \xi |x-y|$.

Subnetworks and Subinstances. Given an instance $\mathcal{G} = (G(V, E), (c_e)_{e \in E}, r)$, any subgraph $H(V, E')$, $E' \subseteq E$, obtained from G by edge deletions, is a *subnetwork* of G. H has the same origin s and destination t as G, and its edges have the same latency functions as in $\mathcal{G}$. Each instance $\mathcal{H} = (H(V, E'), (c_e)_{e \in E'}, r)$, where $H(V, E')$ is a subnetwork of $G(V, E)$, is 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 and every edge e, we let $f_e = \sum_{p: e \in p} f_p$ denote the amount of flow that f routes through e. A path p (resp. edge e) is used by flow f if $f_p > 0$ (resp. $f_e > 0$). Given a flow f, the latency of each edge e is $c_e(f_e)$, and the *bottleneck cost* of each path p is $b_p(f) = \max_{e \in p} c_e(f_e)$. The *bottleneck cost* of a flow f, denoted $B(f)$, is $B(f) = \max_{p: f_p > 0} b_p(f)$. An *optimal* flow of $\mathcal{G}$, denoted o, minimizes the bottleneck cost among all $\mathcal{G}$-feasible flows. We let $B^*(\mathcal{G}) = B(o)$. We note that for every subinstance $\mathcal{H}$ of $\mathcal{G}$, $B^*(\mathcal{H}) \geq B^*(\mathcal{G})$.

Nash Flows and their Properties. A flow f is at *Nash equilibrium*, or simply, is a *Nash flow*, if f routes all traffic on paths of a locally minimum bottleneck cost. Formally, f is a Nash flow if for all $p, p' \in \mathcal{P}$, if $f_p > 0$, then $b_p(f) \leq b_{p'}(f)$. Therefore, in a Nash flow f, all players incur a common bottleneck cost $B(f) = \min_p b_p(f)$, and for every $s - t$ path p', $B(f) \leq b'_p(f)$.

We observe that if a flow f is a Nash flow for an $s - t$ network $G(V, E)$, then the set of edges e with $c_e(f_e) \geq B(f)$ comprises an $s - t$ cut in G. For the converse, if for some flow f, there is an $s - t$ cut consisting of edges e either with $f_e > 0$ and $c_e(f_e) = B(f)$, or with $f_e = 0$ and $c_e(f_e) \geq B(f)$, then f is a Nash flow. Moreover, for all bottleneck routing games with linear latencies $a_e x$, a flow f is a Nash flow iff the set of edges e with $c_e(f_e) = B(f)$ comprises an $s - t$ cut.

It can be shown that every bottleneck routing game admits at least one Nash flow (see e.g., [7, Proposition 2]), and that there is an optimal flow that is also a Nash flow (see e.g., [3, Corollary 2]). In general, a bottleneck routing game admits many different Nash flows, each with a possibly different bottleneck cost. Given an instance $\mathcal{G}$, we let $B(\mathcal{G})$ denote the bottleneck cost of the players in the worst Nash flow of $\mathcal{G}$, i.e. the Nash flow f that maximizes $B(f)$ among all Nash flows. We refer to $B(\mathcal{G})$ as the worst equilibrium bottleneck cost of $\mathcal{G}$. For convenience, for an instance $\mathcal{G} = (G, c, r)$, we sometimes write $B(G, r)$, instead of $B(\mathcal{G})$, to denote the worst equilibrium bottleneck cost of $\mathcal{G}$. We note that for every subinstance $\mathcal{H}$ of $\mathcal{G}$, $B^*(\mathcal{G}) \leq B(\mathcal{H})$, and that there may be subinstances $\mathcal{H}$ with $B(\mathcal{H}) < B(\mathcal{G})$, which is the essence of Braess's paradox (see e.g., Fig. 1).

Subpath-Optimal Nash Flows. For a flow f and any vertex u, let $b_f(u)$ denote the minimum bottleneck cost of f among all $s - u$ paths. The flow f is a *subpath-optimal Nash flow* [7] if for any vertex u and any $s - t$ path p with $f_p > 0$ that includes u, the bottleneck cost of the $s - u$ part of p is $b_f(u)$. For example, the Nash flow f in Fig. 1.a is not subpath-optimal, because $b_f(v) = 0$, through the edge (s, v), while the bottleneck cost of the path (s, u, v) is 1. For this instance, the only subpath-optimal Nash flow is the optimal flow.

ε-Nash Flows. The definition of a Nash flow can be generalized to that of an "almost Nash" flow: For some constant $\varepsilon > 0$, a flow f is an ε-Nash flow if for all $s - t$ paths p, p', if $f_p > 0$, $b_p(f) \leq b_{p'}(f) + \varepsilon$.

Price of Anarchy. The *Price of Anarchy* (PoA) of an instance $\mathcal{G}$, denoted $\rho(\mathcal{G})$, is the ratio of the worst equilibrium bottleneck cost of $\mathcal{G}$ to the optimal bottleneck cost. Formally, $\rho(\mathcal{G}) = B(\mathcal{G})/B^*(\mathcal{G})$.

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

Best Subnetwork. Given an instance $\mathcal{G} = (G, c, r)$, the *best subnetwork* H^* of G minimizes the worst equilibrium bottleneck cost, i.e., for all subnetworks H of G, $B(H^*, r) \leq B(H, r)$.

Problem Definitions. Next, we study the complexity and the approximability of two basic selfish network design problems for bottleneck routing games:

- **Paradox-Ridden Recognition** (ParRidBC) : Given an instance $\mathcal{G}$, decide if $\mathcal{G}$ is paradox-ridden.
- **Best Subnetwork** (BSubNBC) : Given an instance $\mathcal{G}$, find the best subnetwork H^* of G.

The objective function of BSubNBC is the worst equilibrium bottleneck cost $B(H, r)$ of a subnetwork H. Thus, a (polynomial-time) algorithm A achieves an α-approximation for BSubNBC if for all instances $\mathcal{G}$, A returns a subnetwork H with $B(H, r) \leq \alpha B(H^*, r)$. A subtle point is that given a subnetwork H, we do not know how to efficiently compute the worst equilibrium bottleneck cost $B(H, r)$ (see also [2,12]). To deal with this delicate issue, our hardness results use a certain subnetwork structure to identify a good approximation to BSubNBC.

3 Paradox-Free Topologies and Paradox-Free Nash Flows

We start by discussing two interesting cases where Braess's paradox does not occur. We first observe that for any bottleneck routing game $\mathcal{G}$ defined on a series-parallel network, $\rho(\mathcal{G}) = 1$, and thus Braess's paradox does not occur. We recall that a directed $s - t$ network is series-parallel iff it does not contain a θ-graph with degree-2 terminals as a topological minor. Therefore, the example in Fig. 1 shows that series-parallel networks is the largest class of networks for which Braess's paradox does not occur (see also [17] for a similar result for additive routing games). The proof is conceptually similar to that of [8, Lemma 4.1].

Proposition 1. *Let $\mathcal{G}$ be a bottleneck routing game on an $s - t$ series-parallel network. Then, $\rho(\mathcal{G}) = 1$.*

Next, we observe that any subpath-optimal Nash flow achieves an optimal bottleneck cost. Thus, Braess's paradox does not occur if we only consider subpath-optimal Nash flows.

Proposition 2. *Let $\mathcal{G}$ be any bottleneck routing game, and let f be any subpath-optimal Nash flow of $\mathcal{G}$. Then, $B(f) = B^*(\mathcal{G})$.*

4 Recognizing Paradox-Ridden Instances Is Hard

Next, we show that given a linear bottleneck routing game $\mathcal{G}$, it is NP-hard not only to decide whether $\mathcal{G}$ is paradox-ridden, but also to approximate the best subnetwork within a factor less than 4/3. To this end, we employ a reduction from the 2-Directed Disjoint Paths problem (2-DDP), where we are given a directed network D and distinguished vertices s_1, s_2, t_1, t_2, and ask whether D contains a pair of vertex-disjoint paths connecting s_1 to t_1 and s_2 to t_2. 2-DDP is NP-complete, even if the network D is known to contain two edge-disjoint paths connecting s_1 to t_2 and s_2 to t_1 [9, Theorem 3]. In the following, we say that a subnetwork D' of D is *good* if D' contains (i) at least one path outgoing from each of s_1 and s_2 to either t_1 or t_2, (ii) at least one path incoming to each of t_1 and t_2 from either s_1 or s_2, and (iii) either no $s_1 - t_2$ paths or no $s_2 - t_1$ paths. We say that D' is *bad* if any of these conditions is violated by D'. We note that we can efficiently check whether a subnetwork D' of D is good, and that a good subnetwork D' serves as a certificate that D is a YES-instance of 2-DDP. The following lemma directly implies the hardness result of this section.

Lemma 1. *Let $\mathcal{I} = (D, s_1, s_2, t_1, t_2)$ be any* 2-DDP *instance. Then, we can construct, in polynomial time, an $s - t$ network $G(V, E)$ with a linear latency function $c_e(x) = a_e x$, $a_e > 0$, on each edge e, so that for any traffic rate $r > 0$, the bottleneck routing game $\mathcal{G} = (G, c, r)$ has $B^*(\mathcal{G}) = r/4$, and:*

1. *If $\mathcal{I}$ is a* YES*-instance of* 2-DDP*, there exists a subnetwork H of G with $B(H, r) = r/4$.*
2. *If $\mathcal{I}$ is a* NO*-instance of* 2-DDP*, for all subnetworks H' of G, $B(H', r) \geq r/3$.*
3. *For all subnetworks H' of G, either H' contains a good subnetwork of D, or $B(H', r) \geq r/3$.*

Proof sketch. We construct the network G by adding 4 vertices, s, t, v, u, to D and 9 "external" edges $e_1 = (s, u)$, $e_2 = (u, v)$, $e_3 = (v, t)$, $e_4 = (s, v)$, $e_5 = (v, s_1)$, $e_6 = (s, s_2)$, $e_7 = (t_1, u)$, $e_8 = (u, t)$, $e_9 = (t_2, t)$ (see also Fig. 2.a). The external edges e_1 and e_3 have latency $c_{e_1}(x) = c_{e_3}(x) = x/2$. The external edges $e_4, \ldots, e_9$ have latency $c_{e_i} = x$. The external edge e_2 and each edge e of D have latency $c_{e_2}(x) = c_e(x) = \varepsilon x$, for some $\varepsilon \in (0, 1/4)$.

We first observe that $B^*(\mathcal{G}) = r/4$. As for (1), by hypothesis, there are vertex-disjoint paths in D, p and q, connecting s_1 to t_1, and s_2 to t_2. Let H be the subnetwork of G that includes all external edges and only the edges of p and q from D (see also Fig. 2.b). We let $\mathcal{H} = (H, c, r)$ be the corresponding subinstance of $\mathcal{G}$. The flow routing $r/4$ units through each of the paths (e_4, e_5, p, e_7, e_8) and (e_6, q, e_9), and $r/2$ units through (e_1, e_2, e_3), is an $\mathcal{H}$-feasible Nash flow with a bottleneck cost of $r/4$.

We proceed to show that any Nash flow of $\mathcal{H}$ achieves a bottleneck cost of $r/4$. For sake of contradiction, let f be a Nash flow of $\mathcal{H}$ with $B(f) > r/4$. Since f is a Nash flow, the edges e with $c_e(f_e) \geq B(f)$ form an $s - t$ cut in H. Since the bottleneck cost of e_2 and of any edge in p and q is at most $r/4$, this cut includes either e_6 or e_9 (or both), either e_1 or e_3 (or both), and either e_4 or e_8

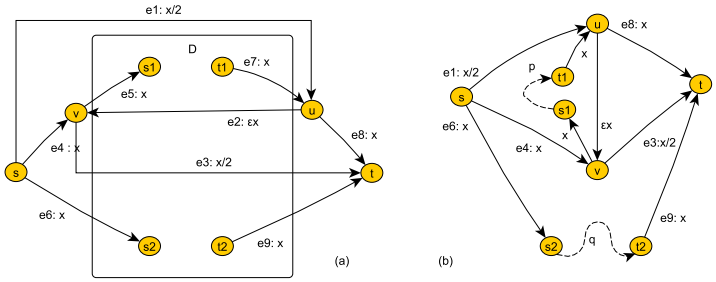

Fig. 2. (a) The network G constructed in the proof of Lemma 1. (b) The best subnetwork of G, with PoA $= 1$, for the case where D contains a pair of vertex-disjoint paths connecting s_1 to t_1 and s_2 to t_2.

(or e_5 or e_6, in certain combinations with other edges). Let us consider the case where this cut includes e_1, e_4, and e_6. Since the bottleneck cost of these edges is greater than $r/4$, we have more than $r/2$ units of flow through e_1 and more than $r/4$ units of flow through each of e_4 and e_6. Hence, we obtain that more than r units of flow leave s, a contradiction. All other cases are similar.

To conclude the proof, we first observe that (3) implies (2), because if $\mathcal{I}$ is a NO-instance, any two paths, p and q, connecting s_1 to t_1 and s_2 to t_2, have some vertex in common, and thus, D does not include any good subnetworks.

To sketch the proof of (3), we let H' be any subnetwork of G, and let $\mathcal{H}'$ be the corresponding subinstance of $\mathcal{G}$. We can show that either H' contains (i) all external edges, (ii) at least one path outgoing from each of s_1 and s_2 to either t_1 or t_2, and (iii) at least one path incoming to each of t_1 and t_2 from either s_1 or s_2, or H' includes a "small" $s-t$ cut, and any $\mathcal{H}'$-feasible flow f has $B(f) \geq r/3$.

Let us now consider a subnetwork H' of G that does not contain a good subnetwork of D, but it satisfies (i), (ii), and (iii) above. By (ii) and (iii), and the hypothesis that the subnetwork of D included in H' is bad, H' contains an $s_1 - t_2$ path p and an $s_2 - t_1$ path q. At the intuitive level, this corresponds to the case where no edges are removed from G. Then, routing $r/3$ units of flow on each of the $s - t$ paths (e_1, e_2, e_3), (e_1, e_2, e_5, p, e_9), and (e_6, q, e_7, e_2, e_3) has a bottleneck cost of $r/3$ and is a Nash flow, because the edges with bottleneck cost $r/3$ comprise an $s - t$ cut. □

The bottleneck routing game $\mathcal{G}$, in Lemma 1, has $\rho(\mathcal{G}) = 4/3$, and is paradox-ridden, if $\mathcal{I}$ is a YES instance of 2-DDP, and paradox-free, otherwise. Hence:

Theorem 1. *Deciding whether a bottleneck routing game with strictly increasing linear latencies is paradox-ridden is* NP*-hard.*

Moreover, Lemma 1 implies that it is NP-hard to approximate BSubNBC within a factor less than $4/3$. A subtle point here is that given a subnetwork H, we do not know how to efficiently compute the worst equilibrium bottleneck cost $B(H, r)$. However, we can use the notion of a good subnetwork of D, and deal with this issue (see also the discussion before Theorem 2).

5 Approximating the Best Subnetwork Is Hard

Next, we recursively apply the construction of Lemma 1, and show that it is NP-hard to approximate BSubNBC within a factor of $O(n^{.121-\varepsilon})$, for any $\varepsilon > 0$.

We consider an $s-t$ network G that can be constructed in polynomial time from a 2-DDP instance $\mathcal{I}$, and includes (possibly many copies of) D. G has a linear latency $c_e(x) = a_e x$ on each edge e, and for any rate $r > 0$, the bottleneck routing game $\mathcal{G} = (G, c, r)$ has $B^*(\mathcal{G}) = r/\gamma_1$, for some $\gamma_1 > 0$. Moreover,

1. If $\mathcal{I}$ is a YES-instance, there exists a subnetwork H of G with $B(H, r) = r/\gamma_1$.
2. If $\mathcal{I}$ is a NO-instance, for all subnetworks H' of G, $B(H', r) \geq r/\gamma_2$, for some $\gamma_2 \in (0, \gamma_1)$.
3. For all subnetworks H' of G, either H' contains at least one copy of a good subnetwork of D, or $B(H', r) \geq r/\gamma_2$.

The existence of such a network G shows that it is NP-hard to approximate BSubNBC within a factor less than $\gamma = \gamma_1/\gamma_2$. Thus, we refer to G as a γ-gap instance. E.g., the network constructed in the proof of Lemma 1 has $\gamma_1 = 4$ and $\gamma_2 = 3$, and thus it is a 4/3-gap instance. We next show that given $\mathcal{I}$ and a γ_1/γ_2-gap instance G, we can construct a $4\gamma_1/(3\gamma_2)$-gap instance G'.

Lemma 2. *Let $\mathcal{I} = (D, s_1, s_2, t_1, t_2)$ be a* 2-DDP *instance, and let G be a γ_1/γ_2-gap instance with linear latencies, based on $\mathcal{I}$. Then, we can construct, in time polynomial in the size of $\mathcal{I}$ and G, an $s-t$ network G' with a linear latency function $c_e(x) = a_e x$, $a_e > 0$, on each edge e, so that for any traffic rate $r > 0$, the bottleneck routing game $\mathcal{G}' = (G', c, r)$ has $B^*(\mathcal{G}) = r/(4\gamma_1)$, and:*

1. *If $\mathcal{I}$ is a* YES*-instance, there is a subnetwork H of G' with $B(H, r) = r/(4\gamma_1)$.*
2. *If $\mathcal{I}$ is a* NO*-instance, for all subnetworks H', $B(H', r) \geq r/(3\gamma_2)$.*
3. *For all subnetworks H' of G', either H' contains at least one copy of a good subnetwork of D, or $B(H', r) \geq r/(3\gamma_2)$.*

The proof applies the construction of Lemma 1, but with all external edges, except for e_2, replaced by a copy of the gap-instance G. Hence, the number of vertices of G' is at most 8 times the number of vertices of G plus the number of vertices of D. If we start with an instance $\mathcal{I}$ of 2-DDP where D has k vertices, and apply Lemma 1 once, and subsequently apply Lemma 2 for $\lfloor \log_{4/3} k \rfloor$ times, we obtain a k-gap instance $\mathcal{G}'$ where G' has $n = O(k^{8.23})$ vertices. Suppose now that there is a polynomial-time algorithm A that approximates the best subnetwork of G' within a factor of $O(k^{1-\varepsilon}) = O(n^{0.121-\varepsilon})$, for a constant $\varepsilon > 0$. Then, if $\mathcal{I}$ is a YES-instance, algorithm A, applied to G', should return a best subnetwork H with at least one copy of a good subnetwork of D. Since H contains a polynomial number of copies of subnetworks of D, and we can check this in polynomial time, and efficiently recognize $\mathcal{I}$ as a YES-instance of 2-DDP. On the other hand, if $\mathcal{I}$ is a NO-instance, D includes no good subnetworks. Again, we can efficiently check that in the subnetwork returned by A, there are not any copies of a good subnetwork of D, and hence recognize $\mathcal{I}$ as a NO-instance of 2-DDP. Thus:

Theorem 2. *For bottleneck routing games with linear latencies, it is* NP*-hard to approximate* BSubNBC *within a factor of $O(n^{0.121-\varepsilon})$, for any constant $\varepsilon > 0$.*

6 Networks with Quasipolynomially Many Paths

In this section, we approximate, in quasipolynomial-time, the best subnetwork and its worst equilibrium bottleneck cost for instances $\mathcal{G} = (G, c, r)$ where the network G has quasipolynomially many $s-t$ paths, the latency functions satisfy a Lipschitz condition, and the worst Nash flow in the best subnetwork routes a non-negligible amount of flow on all edges.

The restriction to networks with quasipolynomially many $s-t$ paths is somehow necessary, in the sense that Theorem 2 shows that if the network has exponentially many $s-t$ paths, as it happens for the hard instances of 2-DDP, and thus for the networks G and G' in the proofs of Lemma 1 and Lemma 2, it is NP-hard to approximate BSubNBC within any reasonable factor. In addition, we assume here that there is a constant $\delta > 0$, such that the worst Nash flow in the best subnetwork H^* routes more than δ units of flow on all edges of H^*.

W.l.o.g., we normalize the traffic rate r to 1. Our algorithm is based on [11, Lemma 2], which applies Althöfer's Lemma [1], and shows that any flow can be approximated by a sparse flow using logarithmically many paths.

Lemma 3. *Let $\mathcal{G} = (G(V, E), c, 1)$ be an instance, and let f be a flow. Then, for any $\varepsilon > 0$, there exists a $\mathcal{G}$-feasible flow $\tilde{f}$ using at most $k(\varepsilon) = \lfloor \log(2m)/(2\varepsilon^2) \rfloor + 1$ paths, such that for all edges e, $|\tilde{f}_e - f_e| \leq \varepsilon$, if $f_e > 0$, and $\tilde{f}_e = 0$, otherwise.*

By Lemma 3, there exists a sparse flow $\tilde{f}$ that approximates the worst Nash flow f on the best subnetwork H^* of G. Moreover, the proof of [11, Lemma 2] shows that the flow $\tilde{f}$ is determined by a multiset P of at most $k(\varepsilon)$ paths, selected among the paths used by f. Then, for every path $p \in \mathcal{P}$, $\tilde{f}_p = |P(p)|/|P|$, where $|P(p)|$ is number of times the path p is included in the multiset P. Therefore, if the total number $|\mathcal{P}|$ of $s-t$ paths in G is quasipolynomial, we can find, by exhaustive search, in quasipolynomial-time, a flow-subnetwork pair that approximates the optimal solution of BSubNBC. Based on this intuition, we can obtain an approximation algorithm for BSubNBC on networks with quasipolynomially many paths, under the technical assumption that the worst Nash flow in the best subnetwork routes a non-negligible amount of flow on all edges.

Theorem 3. *Let $\mathcal{G} = (G(V, E), c, 1)$ be a bottleneck routing game with latency functions that satisfy the Lipschitz condition with a constant $\xi > 0$, let H^* be the best subnetwork of G, and let f^* be the worst Nash flow in H^*. If for all edges e of H^*, $f^*_e > \delta$, for some constant $\delta > 0$, then for any constant $\varepsilon > 0$, we can compute in time $|\mathcal{P}|^{O(\log(2m)/\min\{\delta^2, \varepsilon^2/\xi^2\})}$ a flow f and a subnetwork H such that: (i) f is an $\varepsilon/2$-Nash flow in the subnetwork H, (ii) $B(f) \leq B(H^*, 1) + \varepsilon$, (iii) $B(H, 1) \leq B(f) + \varepsilon/4$, and (iv) $B(f) \leq B(H, 1) + \varepsilon/2$.*

The algorithm of Theorem 3 computes a flow-subnetwork pair (H, f) such that f is an $\varepsilon/2$-Nash flow in H, the worst equilibrium bottleneck cost of H approximates the worst equilibrium bottleneck cost of H^*, since $B(H^*, 1) \leq B(H, 1) \leq B(H^*, 1) + 5\varepsilon/4$, by (ii) and (iii), and the bottleneck cost of f approximates the worst equilibrium bottleneck cost of H, since $B(H, 1) - \varepsilon/4 \leq B(f) \leq B(H, 1) + \varepsilon/2$, by (iii) and (iv).

References

1. Althöfer, I.: On sparse approximations to randomized strategies and convex combinations. Linear Algebra and Applications 99, 339–355 (1994)
2. 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)
3. Banner, R., Orda, A.: Bottleneck routing games in communication networks. IEEE Journal on Selected Areas in Communications 25(6), 1173–1179 (2007)
4. Braess, D.: Über ein paradox aus der Verkehrsplanung. Unternehmensforschung 12, 258–268 (1968)
5. Busch, C., Magdon-Ismail, M.: Atomic routing games on maximum congestion. Theoretical Computer Science 410, 3337–3347 (2009)
6. Caragiannis, I., Galdi, C., Kaklamanis, C.: Network Load Games. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 809–818. Springer, Heidelberg (2005)
7. Cole, R., Dodis, Y., Roughgarden, T.: Bottleneck links, variable demand, and the tragedy of the commons. In: Proc. of the 17th ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, pp. 668–677 (2006)
8. Epstein, A., Feldman, M., Mansour, Y.: Efficient graph topologies in network routing games. Games and Economic Behaviour 66(1), 115–125 (2009)
9. Fortune, S., Hopcroft, J.E., Wyllie, J.: The directed subgraph homeomorphism problem. Theoretical Computer Science 10, 111–121 (1980)
10. Fotakis, D., Kaporis, A.C., Lianeas, T., Spirakis, P.: On the hardness of network design for bottleneck routing games. CoRR, abs/1207.5212 (2012)
11. Fotakis, D., Kaporis, A.C., Spirakis, P.G.: Efficient Methods for Selfish Network Design. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part II. LNCS, vol. 5556, pp. 459–471. Springer, Heidelberg (2009)
12. Hou, H., Zhang, G.: The Hardness of Selective Network Design for Bottleneck Routing Games. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 58–66. Springer, Heidelberg (2007)
13. Koutsoupias, E., Papadimitriou, C.: Worst-Case Equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
14. Lin, H., Roughgarden, T., Tardos, É.: A stronger bound on Braess's paradox. In: Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, pp. 340–341 (2004)
15. 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)
16. Mazalov, V., Monien, B., Schoppmann, F., Tiemann, K.: Wardrop Equilibria and Price of Stability for Bottleneck Games with Splittable Traffic. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 331–342. Springer, Heidelberg (2006)
17. Milchtaich, I.: Network topology and the efficiency of equilibrium. Games and Economic Behavior 57, 321–346 (2006)
18. Roughgarden, T.: 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)