eCOMPASS

eCOMPASS – TR – 004

On the Hardness of Network Design for Bottleneck Routing Games

Dimitris Fotakis, Alexis C. Kaporis, Thanasis Lianeas, Paul G. Spirakis

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, Greece.
[3] Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece.
[4] Research Academic Computer Technology Institute, N. Kazantzaki Str., University Campus, 26500 Patras, Greece.
Email: fotakis@cs.ntua.gr, kaporisa@gmail.com, tlianeas@mail.ntua.gr, spirakis@cti.gr

Abstract. In routing games, the selfish behavior of the players may lead to a degradation of the network performance at equilibrium. In more than a few cases however, the equilibrium performance can be significantly improved if we remove some edges from the network. This counterintuitive fact, widely known as Braess's paradox, gives rise to the (selfish) network design problem, where we seek to recognize routing games suffering from the paradox, and to improve their equilibrium performance by edge removal. In this work, we investigate the computational complexity and the approximability of the network design problem 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, i.e. the maximum latency of a used edge. 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 cannot be improved by edge removal, even if their PoA is as large as $\Omega(n^{0.121})$. This implies that the network design problem for linear bottleneck routing games is NP-hard to approximate within a factor of $O(n^{0.121-\varepsilon})$, for any constant $\varepsilon > 0$. The proof is based on a recursive construction of hard instances that carefully exploits the properties of bottleneck routing games, and may be of independent interest. On the positive side, we 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 used edges. We show that the running time is essentially determined by the total number of paths in the network, and is quasipolynomial when the number of paths is quasipolynomial.

1 Introduction

An typical instance of a non-atomic *bottleneck routing game* consists of a directed network, with an origin s and a destination t, where each edge is associated with a non-decreasing function that determines the edge's latency as a function of its traffic. A rate of traffic is controlled by an infinite population of players, each willing to route a negligible amount of traffic through an $s - t$ path. The players are non-cooperative and selfish, and 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 in communication networks (see e.g., [6, 3] and the references therein).

Previous Work and Motivation. Every bottleneck routing game is known to 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]). On the other hand, bottleneck routing games usually admit many different 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' non-cooperative and selfish behavior in (several variants of) bottleneck routing games. This is typically measured by the *Price of Anarchy* (PoA) [12], which 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, Figure 2]) demonstrate that the PoA of bottleneck routing games with linear latency functions can be as large as $\Omega(n)$, where n is the number of vertices of the network. For atomic splittable bottleneck routing games, where the population of players is finite, and each player controls 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. [15] 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 latency functions of degree d, the PoA is $O(m^d)$, where m is the number of edges of the network. On the other hand, Epstein, Feldman, and Mansour [8] proved that for series-parallel networks with arbitrary latency functions, all Nash flows are optimal. Subsequently, 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 bounded from above by $O(l + \log n)$, where l is the length of the longest $s - t$ path, and by $O(k^2 + \log^2 n)$, where k is length of the longest circuit.

With the PoA of bottleneck routing games so high and crucially depending on topological properties of the network, a natural approach to improving the performance at equilibrium is to exploit the essence of 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
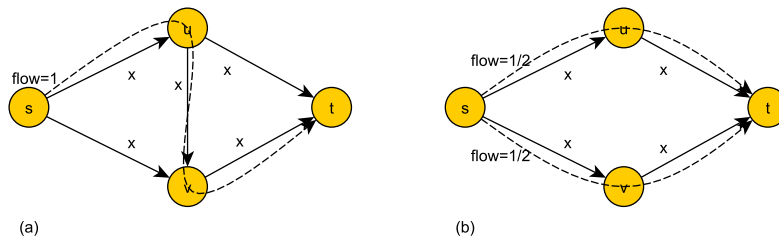
Fig. 1. An example of Braess's paradox for bottleneck routing games. We consider a routing instance with identity latency functions and a unit of traffic to be routed from s to t. The worst Nash flow, in (a), routes all flow through the path (s, u, v, t), and has a bottleneck cost of 1. On the other hand, the optimal flow routes $1/2$ unit through the path (s, u, t) and $1/2$ unit through the path (s, v, t), and has a bottleneck cost of $1/2$. Hence, PoA $= 2$. In the subnetwork (b), obtained by removing the edge (u, v), we have a unique Nash flow that coincides with the optimal flow, and thus the PoA becomes 1. Hence the network on the left is *paradox-ridden*, and the network on the right is the *best subnetwork* of it.

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 classes of instances: *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 [17, 10]).

The approximability of selective network design, a generalization of network design where we cannot remove certain edges, was considered by Hou and Zhang [11]. 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 $\mathrm{O}(m^{d-\varepsilon})$, for any constant $\varepsilon > 0$. 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 [11] 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 high 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. [13] 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. [14] presented an instance with two origin-destination pairs where the removal of a single edge improves the 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 [11] 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-destination pair for all players.

Contribution. Hence, 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 the special case of strictly increasing linear latencies. To the best of our knowledge, this is the first work that investigates the impact of Braess's paradox and the approximability of the network design problem for the basic variant of bottleneck routing games.

2

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. In particular, 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 $\mathrm{O}(n^{0.121-\varepsilon})$, 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 latency functions that satisfy a Lipschitz condition, 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 used 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 [10]. 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 $\mathrm{O}(\varepsilon)$ 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.

Other Related Work. Considerable attention has been paid to the approximability of the network design problem 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. In fact, Roughgarden [17] first introduced the selfish network design problem in this setting, and proved that it is NP-hard to recognize paradox-ridden instances. Roughgarden also proved that it is NP-hard to approximate the network design problem for additive routing 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 [16] proved that non-atomic additive routing games on series-parallel networks do not suffer from Braess's paradox. Fotakis, Kaporis, and Spirakis [10] 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 an origin s and a destination t, $c_e : [0, r] \mapsto \mathbb{R}_{\geq 0}$ is a continuous non-decreasing latency function associated with each edge e, and $r > 0$ is the traffic rate entering at s and leaving at 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 a routing 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 the edges of H 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 each 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 $B(f) = \max_{p : f_p > 0} b_p(f)$, i.e., the maximum bottleneck cost of any used path.

Optimal Flow. An *optimal* flow of an instance $\mathcal{G}$, 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. We consider a non-atomic model of selfish routing, where the traffic is divided among an infinite population of players, each routing a negligible amount of traffic from s to t. 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 $s - t$ paths p, 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 of the players. 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).

The following proposition considers the effect of a uniform scaling of the latency functions. For completeness, we include the proof in the Appendix, Section A.1.

Proposition 1. *Let $\mathcal{G} = (G, c, r)$ be a routing instance, let $\alpha > 0$, and let $\mathcal{G}' = (G, \alpha c, r)$ be the routing instance obtained from $\mathcal{G}$ if we replace the latency function $c_e(x)$ of each edge e with $\alpha c_e(x)$. Then, any $\mathcal{G}$-feasible flow f is also $\mathcal{G}'$-feasible and has $B_{\mathcal{G}'}(f) = \alpha B_{\mathcal{G}}(f)$. Moreover, a flow f is a Nash flow (resp. optimal flow) of $\mathcal{G}$ iff f is a Nash flow (resp. optimal flow) of $\mathcal{G}'$.*

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 with $1/2$ unit on the path (s, u, t) and $1/2$ unit on the path (s, v, t).

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. If an instance is not paradox-free, edge removal can decrease the worst equilibrium bottleneck cost by a factor greater than 1 and at most $\rho(\mathcal{G})$. 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. In this work, we investigate the complexity and the approximability of two fundamental 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.

We note that 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, 11], where a similar issue arises). To deal with this delicate issue, our hardness results use a certain subnetwork structure to identify a good approximation to BSubNBC.

Series-Parallel Networks. A directed $s - t$ network is *series-parallel* if it either consists of a single edge (s, t) or can be obtained from two series-parallel graphs with terminals (s_1, t_1) and (s_2, t_2) composed either in series or in parallel. In a *series composition*, t_1 is identified with s_2, s_1 becomes s, and t_2 becomes t. In a *parallel composition*, s_1 is identified with s_2 and becomes s, and t_1 is identified with t_2 and becomes t.

3 Paradox-Free Network Topologies and Paradox-Free Nash Flows

We start by discussing two interesting cases where Braess's paradox does not occur. We first show that if we have a bottleneck routing game $\mathcal{G}$ defined on an $s - t$ series-parallel network, then $\rho(\mathcal{G}) = 1$, and thus Braess's paradox does not occur. We recall that this was also pointed out in [8] for the case of atomic unsplittable bottleneck routing games. Moreover, we note 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 demonstrates that series-parallel networks is the largest class of network topologies for which Braess's paradox does not occur (see also [16] for a similar result for the case of additive routing games). The proof of the following proposition is conceptually similar to the proof of [8, Lemma 4.1], and can be found in the Appendix, Section A.2.

Proposition 2. *Let $\mathcal{G}$ be bottleneck routing game on an $s-t$ series-parallel network. Then, $\rho(\mathcal{G}) = 1$.*

Next, we show that any subpath-optimal Nash flow achieves a minimum bottleneck cost, and thus Braess's paradox does not occur if we restrict ourselves to subpath-optimal Nash flows.

Proposition 3. *Let $\mathcal{G}$ be bottleneck routing game, and let f be any subpath-optimal Nash flow of $\mathcal{G}$. Then, $B(f) = B^*(\mathcal{G})$.*

Proof. Let f be any subpath-optimal Nash flow of $\mathcal{G}$, let S be the set of vertices reachable from s via edges with bottleneck cost less than $B(f)$, let $\delta^+(S)$ be the set of edges $e = (u, v)$ with $u \in S$ and $v \notin S$, and let $\delta^-(S)$ be the set of edges $e = (u, v)$, with $u \notin S$ and $v \in S$. Then, in [7, Lemma 4.5], it is shown that (i) $(S, V \setminus S)$ is an $s-t$ cut, (ii) for all edges $e \in \delta^+(S)$, $c_e(f_e) \geq B(f)$, (iii) for all edges $e \in \delta^+(S)$ with $f_e > 0$, $c_e(f_e) = B(f)$, and (iv) for all edges $e \in \delta^-(S)$, $f_e = 0$.

By (i) and (iv), any optimal flow o routes at least as much traffic as the subpath-optimal Nash flow f routes through the edges in $\delta^+(S)$. Thus, there is some edge $e \in \delta^+(S)$ with $o_e \geq f_e$, which implies that $c_e(o_e) \geq c_e(f_e) \geq B(f)$, where the second inequality follows from (ii). Since $B^*(\mathcal{G}) = B(o) \geq c_e(o_e)$, we obtain that $B^*(\mathcal{G}) = B(f)$. $\quad\square$

4 Recognizing Paradox-Ridden Instances is Hard

In this section, 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 was shown NP-complete in [9, Theorem 3], 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. 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. Then, 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. We construct a network $G(V, E)$ with the desired properties 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 show that $B^*(\mathcal{G}) = r/4$. As for the lower bound, since the edges e_1, e_4, and e_6 form an $s-t$ cut in G, every $\mathcal{G}$-feasible flow has a bottleneck cost of at least $r/4$. As for the upper bound, we may assume that D contains an $s_1 - t_2$ path p and an $s_2 - t_1$ path q, which are edge-disjoint
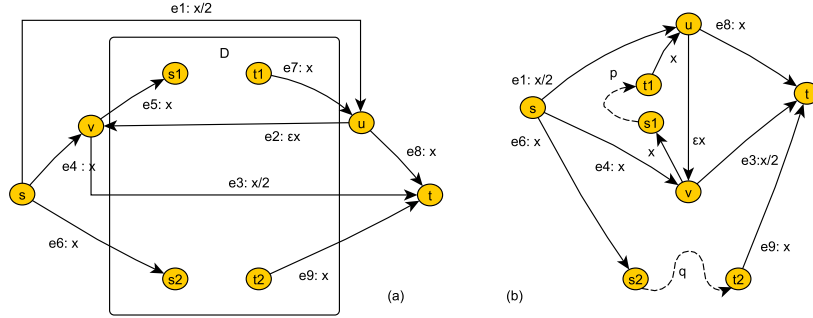
Fig. 2. (a) The network G constructed in the proof of Lemma 1. (b) The best subnetwork of G, with PoA $= 1$, for the case where D contains a pair of vertex-disjoint paths connecting s_1 to t_1 and s_2 to t_2.

(see also [9, Theorem 3]). Then, we route a flow of $r/4$ through each of the paths (e_4, e_5, p, e_9) and (e_6, q, e_7, e_8), and a flow of $r/2$ through the path (e_1, e_2, e_3), which gives a bottleneck cost of $r/4$.

Next, we show (1), namely that if $\mathcal{I}$ is a YES-instance of 2-DDP, then there exists a subnetwork H of G with $B(H, r) = r/4$. By hypothesis, there is a pair of 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 the path (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 (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 have also to show (3), namely that for any subnetwork H' of G, if H' does not contain a good subnetwork of D, then $B(H', r) \geq r/3$. We 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 includes no good subnetworks. To show (3), we let H' be any subnetwork of G, and let $\mathcal{H}'$ be the corresponding subinstance of $\mathcal{G}$. We first 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 thus any $\mathcal{H}'$-feasible flow f has $B(f) \geq r/3$.

To prove (i), we observe that if some of the edges e_1, e_4, and e_6 is missing from H', r units of flow are routed through the remaining ones, which results in a bottleneck cost of at least $r/3$. The same argument applies to the edges e_3, e_8, and e_9. Similarly, if e_2 is not present in H', the edges e_4, e_6, and e_8 form an $s - t$ cut, and routing r units of flow through them causes a bottleneck cost of at least $r/3$. Therefore, we can assume, without loss of generality, that all these external edges are present in H'.

Now, let us focus on the external edges e_5 and e_7. If e_5 is not present in H' and there is a path p outgoing from s_2 to either t_1 or t_2, routing $2r/3$ units of flow through the path (e_1, e_2, e_3) and $r/3$ units through the path (e_6, p, e_9) (or through the path (e_6, p, e_7, e_8)) is a Nash flow with a bottleneck cost of $r/3$ (see also Fig. 3.a). If s_2 is connected to neither t_1 nor t_2 (no matter whether e_5 is present in H' or not), the edges e_1 and e_4 form an $s - t$ cut, and thus, any $\mathcal{H}'$-feasible flow has a bottleneck
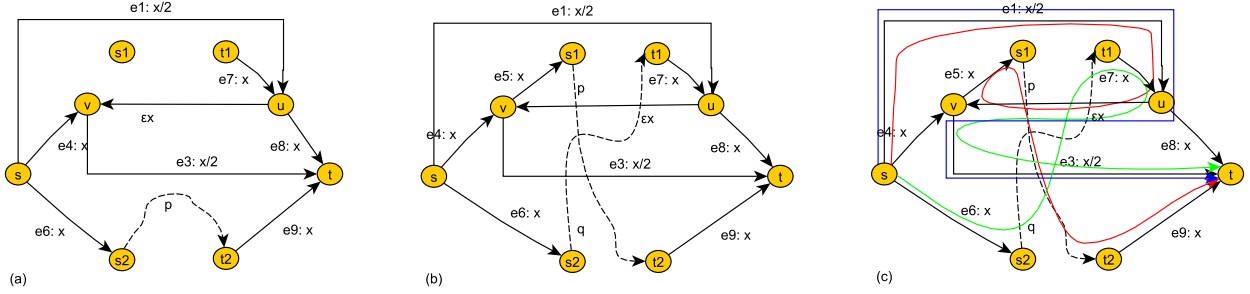
Fig. 3. Possible subnetworks of G when there is no pair of vertex-disjoint paths connecting s_1 to t_1 and s_2 to t_2. The subnetwork (a) contains an $s_2 - t_2$ path and does not include e_5. In the subnetwork (b), we essentially have all edges of G. In (c), we depict a Nash flow that consists of three paths, each carrying $r/3$ units of flow, and has a bottleneck cost of $r/3$.

cost of at least $r/3$. Similarly, we can show that if either e_7 is not present in H', or neither s_1 nor s_2 is connected to t_2, any $\mathcal{H}'$-feasible flow has a bottleneck cost of at least $r/3$. Therefore, we can assume, without loss of generality, that all external edges are present in H', and that H' includes at least one path outgoing from s_2 to either t_1 or t_2, and at least one path incoming to t_2 from either s_1 or s_2.

Similarly, we can assume, without loss of generality, that H' includes at least one path outgoing from s_1 to either t_1 or t_2, and at least one path incoming to t_1 from either s_1 or s_2. E.g., if s_1 is connected to neither t_1 nor t_2, routing $2r/3$ units of flow through the path (e_1, e_2, e_3) and $r/3$ units through s_2 and either t_1 or t_2 (or both) is a Nash flow with a bottleneck cost of $r/3$. A similar argument applies to the case where neither s_1 nor s_2 is connected to t_1.

Let us now consider a subnetwork H' of G that does not contain a good subnetwork of D, but it 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. 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 (see also Fig. 3.b). 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 set of edges with bottleneck cost $r/3$ comprises an $s - t$ cut (see also Fig. 3.c). Therefore, we have shown part (3) of the lemma, which in turn, immediately implies part (2). $\qquad \square$

We note that the bottleneck routing game $\mathcal{G}$ in the proof of 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. Thus, we obtain that:

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 $4/3$. The 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. Specifically, let A be any approximation algorithm for BSubNBC with approximation ratio less than $4/3$. Then, if D is a YES-instance of 2-DDP, A applied to the network G, constructed in the proof of Lemma 1, returns a subnetwork H with $B(H, r) < r/3$. Thus, by Lemma 1, H contains a good subnetwork of D, which can be checked in polynomial time. If D is a NO-instance, D contains no good subnetworks. Hence, the outcome of A would allow us to distinguish between YES and NO instances of 2-DDP.

5 Approximating the Best Subnetwork is Hard

Next, we apply essentially the same construction as in the proof of Lemma 1, but in a recursive way, and show that it is NP-hard to approximate BSubNBC for linear bottleneck routing games within a factor of $O(n^{.121-\varepsilon})$, for any constant $\varepsilon > 0$. Throughout this section, we let $\mathcal{I} = (D, s_1, s_2, t_1, t_2)$ be a 2-DDP instance, and let G be an $s - t$ network, which includes (possibly many copies of) D and can be constructed from $\mathcal{I}$ in polynomial time. We assume that G has a linear latency function $c_e(x) = a_e x$, $a_e > 0$, on each edge e, and for any traffic 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 of 2-DDP, there exists a subnetwork H of G with $B(H, r) = r/\gamma_1$.
2. If $\mathcal{I}$ is a NO-instance of 2-DDP, for all subnetworks H' of G, $B(H', r) \geq r/\gamma_2$, for a $\gamma_2 \in (0, \gamma_1)$.
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 shows that it is NP-hard to approximate BSubNBC within a factor less than $\gamma = \gamma_1/\gamma_2$. Thus, we usually refer to G as a γ-gap instance (with linear latencies). For example, for the network G in the proof of Lemma 1, $\gamma_1 = 4$ and $\gamma_2 = 3$, and thus G 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', i.e., we can amplify the inapproximability gap by a factor of $4/3$.

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

Proof. Starting from D, we obtain G' by applying the construction of Lemma 1, but with all external edges, except for e_2, replaced by a copy of the gap-instance G. For convenience, we refer to the copy of the gap-instance replacing the external edge e_i, $i \in \{1, 3, \ldots, 9\}$, as the *edgework* G_i. Formally, to obtain G', we start from D and add four new vertices, s, t, v, u. We connect s to u, with the $s - u$ edgework G_1, and v to t, with the $s - u$ edgework G_3, where in both G_1 and G_3, we replace the latency function $c_e(x)$ of each edge e in the gap instance with $c_e(x)/2$ (this is because in Lemma 1, the external edges e_1 and e_3 have latencies $x/2$). Moreover, instead of the external edge e_i, $i \in \{4, \ldots, 9\}$, we connect (s, v), (v, s_1), (s, s_2), (t_1, u), (u, t), and (t_2, t) with the edgework G_i. The latencies in these edgeworks are as in the gap instance. Furthermore, we add the external edge $e_2 = (u, v)$ with latency $c_{e_2}(x) = \varepsilon x$, for some $\varepsilon \in (0, \frac{1}{4\gamma_1})$ (see also Fig. 4.a). Also, each edge e of D has latency $c_e(x) = \varepsilon x$. We next consider the corresponding routing instance $\mathcal{G}'$ with an arbitrary traffic rate $r > 0$. Throughout the proof, when we define a routing instance, we omit, for simplicity, the coordinate c, referring to the latency functions, with the understanding that they are defined as above.

Intuitively, each G_i, $i \in \{4, \ldots, 9\}$, behaves as an external edge (hence the term edge(net)work), which at optimality has a bottleneck cost of r/γ_1, for any traffic rate r entering G_i. Moreover, if $\mathcal{I}$ is a YES-instance of 2-DDP, the edgework G_i has a subedgework H_i for which $B(H_i, r) = r/\gamma_1$, for any r, while if H_i does not contain any copies of a good subnetwork of D (or, if $\mathcal{I}$ is a NO-instance), for all subedgeworks H'_i of G_i, $B(H'_i, r) \geq r/\gamma_2$, for any r. The same holds for G_1 and G_3, but with
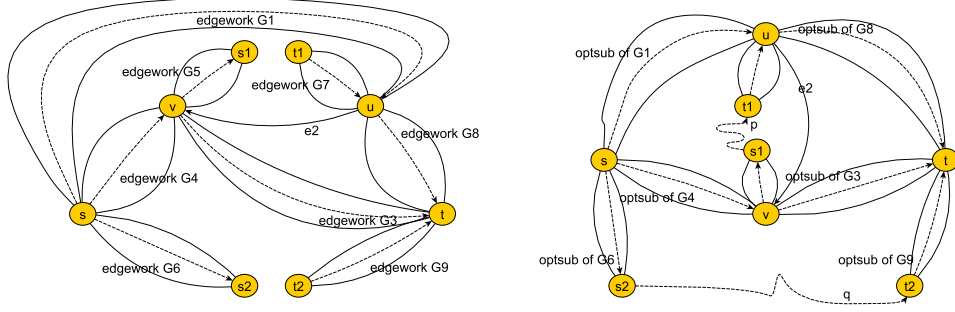
Fig. 4. (a) The network G' constructed in the proof of Lemma 2. The structure of G' is similar to the structure of the network G in Fig. 2, with each external edge e_i, except for e_2, replaced by the edgework G_i. (b) The structure of a best subnetwork H of G', with $\mathrm{PoA} = 1$, when D contains a pair of vertex-disjoint paths, p and q, connecting s_1 to t_1 and s_2 to t_2. To complete H, we use an optimal subnetwork (or simply, subedgework) of each edgework G_i.

a worst equilibrium bottleneck cost of $r/(2\gamma_1)$ in the former case, and of $r/(2\gamma_2)$ in the latter case, because the latency functions of G_1 and G_3 are scaled by $1/2$ (see also Proposition 1).

The proofs of the following propositions are conceptually similar to the proofs of the corresponding claims in the proof Lemma 1, and can be found in the Appendix, Sections A.3 and A.4.

Proposition 4. *The optimal bottleneck cost of $\mathcal{G}'$ is $B^*(\mathcal{G}') = r/(4\gamma_1)$.*

Proposition 5. *If $\mathcal{I}$ is a YES-instance, there is a subnetwork H of G' with $B(H, r) = r/(4\gamma_1)$.*

The most technical part of the proof is to show (3), namely that for any subnetwork H' of G', if H' does not contain any copies of a good subnetwork of D, then $B(H', r) \geq r/(3\gamma_2)$. This immediately implies (2), since if $\mathcal{I}$ is a NO-instance of 2-DDP, D includes no good subnetworks. To prove (3), we consider any subnetwork H' of G', and let H'_i be the subedgework of each G_i present in H'. We assume that the subedgeworks H'_i do not contain any copies of a good subnetwork of D, and show that if the subnetwork of D connecting s_1 and s_2 to t_1 and t_2 in H' is also bad, then $B(H', r) \geq r/(3\gamma_2)$.

At the technical level, we repeatedly use the idea of a flow f_i through a subedgework H'_i that "saturates" H'_i, in the sense that f_i is a Nash flow with bottleneck cost at least $r_i/(3\gamma_2)$ for the subinstance (H'_i, r_i). Formally, we say that a flow rate r_i *saturates* a subedgework H'_i if $B(H'_i, r_i) \geq r_i/(3\gamma_2)$. We refer to the flow rate r_i^s for which $B(H'_i, r_i^s) = r_i^s/(3\gamma_2)$ as the *saturation rate* of H'_i. We note that the saturation rate r_i^s is well-defined, because the latency functions of G_is are linear and strictly increasing. Moreover, by property (3) of gap instances, the saturation rate of each subedgework H'_i is $r_i^s \leq r/3$, if $i \in \{4, \ldots, 9\}$, and $r_i^s \leq 2r/3$, if $i \in \{1, 3\}$. Thus, at the intuitive level, the subedgeworks H'_i behave as the external edges of the network constructed in the proof of Lemma 1. Hence, to show that $B(H', r) \geq r/(3\gamma_2)$, we need to construct a flow of rate (at most) r that saturates a collection of subedgeworks comprising an $s - t$ cut in H'.

Our first step in this direction is to simplify the possible structure of H'. The proof of the following proposition can be found in the Appendix, Section A.5.

Proposition 6. *Let H' be any subnetwork of G' whose subedgeworks H'_i do not contain any copies of a good subnetwork of D. Then, either the subnetwork H' contains (i) the external edge e_2, (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 $B(H', r) \geq r/(3\gamma_2)$.*

Now, let us focus on a subnetwork H' of G' that contains (i) the external edge e_2, (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. If the copy of the subnetwork of D connecting s_1 and s_2 to t_1 and t_2 in H' is also bad, properties (ii) and (iii) imply that H' contains an $s_1 - t_2$ path p and an $s_2 - t_1$ path q. In this case, the entire subnetwork H' essentially behaves as if it included all edges of G'. Then, a routing similar to that in Fig. 3.c gives a Nash flow with a bottleneck cost of $r/(3\gamma_2)$. This intuition is formalized by the following proposition, whose proof can be found in the Appendix, Section A.6.

Proposition 7. *Let H' be any subnetwork of G' that satisfies (i), (ii), and (iii) above, and does not contain any copies of a good subnetwork of D. Then $B(H', r) \geq r/(3\gamma_2)$.*

Propositions 6 and 7 immediately imply part (3) of the lemma, which, in turn, implies part (2). □

Each time we apply Lemma 2 to a γ-gap instance G, we obtain a $4\gamma/3$-gap instance G' with a number of vertices of at most 8 times the vertices of G plus the number of vertices of D. Therefore, if we start with an instance $\mathcal{I} = (D, s_1, s_2, t_1, t_2)$ 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 the network G' has $n = \mathrm{O}(k^{8.23})$ vertices. Suppose now that there is a polynomial-time algorithm A that approximates the best subnetwork of G' within a factor of $\mathrm{O}(k^{1-\varepsilon}) = \mathrm{O}(n^{0.121-\varepsilon})$, for some small $\varepsilon > 0$. Then, if $\mathcal{I}$ is a YES-instance of 2-DDP, 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 whether a subnetwork of D is good in polynomial time, we can efficiently recognize $\mathcal{I}$ as a YES-instance of 2-DDP. On the other hand, if $\mathcal{I}$ is a NO-instance of 2-DDP, D includes no good subnetworks. Again, we can efficiently check that in the subnetwork returned by algorithm 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, we obtain that:

Theorem 2. *For bottleneck routing games with strictly increasing linear latencies, it is NP-hard to approximate BSubNBC within a factor of $\mathrm{O}(n^{0.121-\varepsilon})$, 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 are continuous and satisfy a Lipschitz condition, and the worst Nash flow in the best subnetwork routes a non-negligible amount of flow on all used edges.

We highlight that 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' constructed in the proofs of Lemma 1 and Lemma 2, it is NP-hard to approximate BSubNBC within any reasonable factor. Also, we can always assume, without loss of generality, that the worst Nash flow of the best subnetwork H^* assigns positive flow to all edges of H^*. Otherwise, we can remove any unused edges, without increasing the worst equilibrium bottleneck cost of H^*. In addition, we assume here that there is a constant $\delta > 0$, such that the worst Nash flow in H^* routes more than δ units of flow on all edges of the best subnetwork H^*.

In the following, we normalize the traffic rate r to 1. This is for convenience and can be made without loss of generality[5]. Our algorithm is based on [10, Lemma 2], which applies Althöfer's "Sparsification" Lemma [1], and shows that any flow can be approximated by a "sparse" flow using logarithmically many paths.

---

[5] Given a bottleneck routing game $\mathcal{G}$ with traffic rate $r > 0$, we can replace each latency function $c_e(x)$ with $c_e(rx)$, and obtain a bottleneck routing game $\mathcal{G}'$ with traffic rate 1, and the same Nash flows, PoA, and solutions to BSubNBC.

Lemma 3. *Let $\mathcal{G} = (G(V,E), c, 1)$ be a routing instance, and let f be any $\mathcal{G}$-feasible 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 [10, 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, and $|P|$ is the cardinality of P. Therefore, if the total number $|\mathcal{P}|$ of $s-t$ paths in G is quasipolynomial, we can find, in quasipolynomial-time, by exhaustive search, a flow-subnetwork pair that approximates the optimal solution of BSubNBC. Based on this intuition, we next obtain an approximation algorithm for BSubNBC on networks with quasipolynomially many paths, under the assumption 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^*. This assumption is necessary so that the exhaustive search on the family of sparse flows of Lemma 3 can generate the best subnetwork H^*, which is crucial for the analysis.

Theorem 3. *Let $\mathcal{G} = (G(V,E), c, 1)$ be a bottleneck routing game with continuous 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) $B(f) \leq B(H, 1) + \varepsilon/2$.*

Proof. Let $\varepsilon > 0$ be a constant, and let $\epsilon_1 = \min\{\delta, \varepsilon/(4\xi)\}$, and $\epsilon_2 = \varepsilon/2$. We show that a flow-subnetwork pair (H, f) with the desired properties can be computed in time $|\mathcal{P}|^{O(k(\epsilon_1))}$, where $k(\varepsilon_1) = \lfloor \log(2m)/\min\{2\delta^2, \varepsilon^2/(8\xi^2)\} \rfloor + 1$, For convenience, we say that a flow g is a *candidate flow* if there is a multiset P of paths from $\mathcal{P}$, with $|P| \leq k(\epsilon_1)$, such that $g_p = |P(p)|/|P|$, for each $p \in \mathcal{P}$. Namely, a candidate flow belongs to the family of sparse flows, which by Lemma 3, can approximate any other flow. Similarly, a subnetwork H is a *candidate subnetwork* if there is a candidate flow g such that H consists of the edges used by g (and only of them), and a subnetwork-flow pair (H, g) is a *candidate solution*, if g is a candidate flow, H is a candidate subnetwork that includes all the edges used by g (and possibly some other edges), and g is an ϵ_2-Nash flow in H.

By exhaustive search, in time $|\mathcal{P}|^{O(k(\epsilon_1))}$, we generate all candidate flows, all candidate subnetworks, and compute the bottleneck cost $B(g)$ of any candidate flow g. Then, for each pair (H, g), where g is a candidate flow and H is a candidate subnetwork, we check, in polynomial time, whether g is an ϵ_2-Nash flow in H, and thus whether (H, g) is a candidate solution. Thus, in time $|\mathcal{P}|^{O(k(\epsilon_1))}$, we determine all candidate solutions. For each candidate subnetwork H that participates in at least one candidate solution, we let $\tilde{B}(H)$ be the maximum bottleneck cost $B(g)$ of a candidate flow g for which (H, g) is a candidate solution. The algorithm returns the subnetwork H that minimizes $\tilde{B}(H)$, and a flow f for which (H, f) is a candidate solution and $\tilde{B}(H) = B(f)$.

The exhaustive search above can be implemented in $|\mathcal{P}|^{O(k(\epsilon_1))}$ time. As for the properties of the solution (H, f), the definition of candidate solutions immediately implies (i), i.e., that f is an $\varepsilon/2$-Nash flow in H. In the Appendix, Section A.7, we use Lemma 3, and show (ii), (iii), and (iv). □

Therefore, the algorithm of Theorem 3 returns a flow-subnetwork pair (H, f) such that f is an $\varepsilon/2$-Nash flow in H, the worst equilibrium bottleneck cost of the subnetwork 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).

12

References

1. I. Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and Applications*, 99:339-355, 1994.
2. 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)*, LNCS 3879, pp. 41-54, 2005.
3. R. Banner and A. Orda. Bottleneck routing games in communication networks. *IEEE Journal on Selected Areas in Communications*, 25(6):1173-1179, 2007.
4. D. Braess. Über ein paradox aus der Verkehrsplanung. *Unternehmensforschung*, 12:258-268, 1968.
5. C. Busch and M. Magdon-Ismail. Atomic routing games on maximum congestion. *Theoretical Computer Science*, 410:3337-3347, 2009.
6. I. Caragiannis, C. Galdi, and C. Kaklamanis. Network load games. In *Proc. of the 16th Symp. on Algorithms and Computation (ISAAC '05)*, LNCS 3827, pp. 809-818, 2005.
7. R. Cole, Y. Dodis, and T. Roughgarden. Bottleneck links, variable demand, and the tragedy of the commons. In *Proc. of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA '06)*, pp. 668-677, 2006.
8. A. Epstein, M. Feldman, and Y. Mansour. Efficient graph topologies in network routing games. *Games and Economic Behaviour*, 66(1):115125, 2009.
9. S. Fortune, J.E. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111-121, 1980.
10. D. Fotakis, A.C. Kaporis, and P.G. Spirakis. Efficient methods for selfish network design. In *Proc. of the 36th Colloquium on Automata, Languages and Programming (ICALP-C '09)*, LNCS 5556, pp. 459-471, 2009.
11. H. Hou and G. Zhang. The hardness of selective network design for bottleneck routing games. In *Proc. of the 4th Conference on Theory and Applications of Models of Computation (TAMC '07)*, LNCS 4484, pp. 58-66, 2007.
12. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proc. of the 16th Symposium on Theoretical Aspects of Computer Science (STACS '99)*, LNCS 1563, pp. 404-413, 1999.
13. H. Lin, T. Roughgarden, and É. Tardos. A stronger bound on Braess's paradox. In *Proc. of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*, pp. 340-341, 2004.
14. H. Lin, T. Roughgarden, É. Tardos, and A. Walkover. Braess's paradox, Fibonacci numbers, and exponential inapproximability. In *Proc. of the 32th Colloquium on Automata, Languages and Programming (ICALP '05)*, LNCS 3580, pp. 497-512, 2005.
15. V. Mazalov, B. Monien, F. Schoppmann, and K. Tiemann. Wardrop equilibria and price of stability for bottleneck games with splittable traffic. In *Proc. of the 2nd Workshop on Internet and Network Economics (WINE '06)*, LNCS 4286, pp. 331-342, 2006.
16. I. Milchtaich. Network topology and the efficiency of equilibrium. *Games and Economic Behavior*, 57:321346, 2006.
17. 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.

A Appendix

A.1 The Proof of Proposition 1

Since the traffic rate of both $\mathcal{G}$ and $\mathcal{G}'$ is r, any $\mathcal{G}$-feasible flow f is also $\mathcal{G}'$-feasible. Moreover, the $\mathcal{G}'$-latency of f on each edge e is $\alpha c_e(f_e)$. This immediately implies that $B_{\mathcal{G}'}(f) = \alpha B_{\mathcal{G}}(f)$, and that f is a Nash flow (resp. optimal flow) of $\mathcal{G}$ iff f is a Nash flow (resp. optimal flow) of $\mathcal{G}'$. □

A.2 The Proof of Proposition 1

Let f be any Nash flow of $\mathcal{G}$. We use induction on the series-parallel structure of the network G, and show that f is an optimal flow w.r.t the bottleneck cost, i.e., that $B(f) = B^*(\mathcal{G})$. For the basis, we observe that the claim holds if G consists of a single edge (s, t). For the inductive step, we distinguish two cases, depending on whether G is obtained by the series or the parallel composition of two series-parallel networks G_1 and G_2.

Series Composition. First, we consider the case where G is obtained by the series composition of an $s - t'$ series-parallel network G_1 and a $t' - t$ series-parallel network G_2. We let f_1 and f_2, both of rate r, be the restrictions of f into G_1 and G_2, respectively.

We start with the case where $B(f) = B(f_1) = B(f_2)$. Then, either f_1 is a Nash flow in G_1, or f_2 is a Nash flow in G_2. Otherwise, there would be an $s - t'$ path p_1 in G_1 with bottleneck cost $b_{p_1}(f_1) < B(f_1)$, and an $t' - t$ path p_2 in G_2, with bottleneck cost $b_{p_2}(f_2) < B(f_2)$. Combining p_1 and p_2, we obtain an $s - t$ path $p = p_1 \cup p_2$ in G with bottleneck cost smaller than $B(f)$, which contradicts the hypothesis that f is a Nash flow of $\mathcal{G}$. If f_1 (or f_2) is a Nash flow in G_1 (resp. G_2), then by induction hypothesis f_1 (resp. f_2) is an optimal flow in G_1 (resp. in G_2), and thus f is an optimal flow of $\mathcal{G}$.

Otherwise, we assume, without loss of generality, that $B(f) = B(f_1) < B(f_2)$. Then, f_1 is a Nash flow in G_1. Otherwise, there would be an $s - t'$ path p_1 in G_1 with bottleneck cost $b_{p_1}(f_1) < B(f_1)$, which could be combined with any $t' - t$ path p_2 in G_2, with bottleneck cost $B(f_2) < B(f_1)$, into an $s - t$ path $p = p_1 \cup p_2$ with bottleneck cost smaller than $B(f)$. The existence of such a path p contradicts the the hypothesis that f is a Nash flow of $\mathcal{G}$. Therefore, by induction hypothesis f_1 is an optimal flow in G_1, and thus f is an optimal flow of $\mathcal{G}$.

Parallel Composition. Next, we consider the case where G is obtained by the parallel composition of an $s - t$ series-parallel network G_1 and an $s - t$ series-parallel network G_2. We let f_1 and f_2 be the restriction of f into G_1 and G_2, respectively, let r_1 (resp. r_2) be the rate of f_1 (resp. f_2), and let $\mathcal{G}_1$ (resp. $\mathcal{G}_2$) be the corresponding routing instance. Then, since f is a Nash flow of $\mathcal{G}$, f_1 and f_2 are Nash flows of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, and $B(f_1) = B(f_2) = B(f)$. Therefore, by the induction hypothesis, f_1 and f_2 are optimal flows of $\mathcal{G}_1$ and $\mathcal{G}_2$, and f is an optimal flow of $\mathcal{G}$. To see this, we observe that any flow different from f must route more flow through either G_1 or G_2. But if the flow through e.g. G_1 is more than r_1, the bottleneck cost through G_1 would be at least as large as $B(f_1)$. $\qquad\square$

A.3   The Proof of Proposition 4

We have to show that $B^*(\mathcal{G}') = r/(4\gamma_1)$. For the upper bound, as in the proof of Lemma 1, we assume that D contains an $s_1 - t_2$ path p and an $s_2 - t_1$ path q, which are edge-disjoint. We route (i) $r/4$ units of flow through the edgeworks G_4, G_5, next through the path p, and next through the edgework G_9, (ii) $r/4$ units through the edgeworks G_6, next through the path q, and next through the edgeworks G_7 and G_8, and (ii) $r/2$ units through the edgework G_1, next through the external edge e_2, and next through the edgework G_3. These routes are edge(work)-disjoint, and if we route the flow optimally through each edgework, the bottleneck cost is $r/(4\gamma_1)$. As for the lower bound, we observe that the edgeworks H_1, H_4, and H_6 essentially form an $s - t$ cut in G', and thus every feasible flow has a bottleneck cost of at least $r/(4\gamma_1)$. $\qquad\square$

A.4   The Proof of Proposition 5

If $\mathcal{I}$ is a YES-instance of 2-DDP, then (i) there are two vertex-disjoint paths in D, p and q, connecting s_1 to t_1 and s_2 to t_2, and (ii) there is an optimal subnetwork (or simply, subedgework) H_i of each edgework G_i so that for any traffic rate r routed through H_i, the worst equilibrium bottleneck cost $B(H_i, r)$ is r/γ_1, if $i \in \{4, \ldots, 9\}$, and $r/(2\gamma_1)$, if $i \in \{1, 3\}$. Let H be the subnetwork of G' that consists of only the edges of the paths p and q from D, of the external edge e_2, and of the optimal subedgeworks H_i, $i \in \{1, 3, \ldots, 9\}$ (see also Fig. 4.b). We observe that we can route: (i) $r/4$ units of flow through the subedgeworks H_4, H_5, next through the path p, and next through the subedgeworks H_7 and H_8, (ii) $r/4$ units of flow through the subedgework H_6, next through the path q, and next through the subedgework H_9, and (iii) $r/2$ units of flow through the subedgework H_1, next through the external edge e_2, and next through the subedgework H_3. These routes are edge(work)-disjoint, and

if we use any Nash flow through each of the routing instances $(H_i, r/4)$, $i \in \{4, \ldots, 9\}$, $(H_1, r/2)$, and $(H_3, r/2)$, we obtain a Nash flow of the instance (H, r) with a bottleneck cost of $r/(4\gamma_1)$.

We next show that any Nash flow of (H, r) has a bottleneck cost of at most $r/(4\gamma_1)$. To reach a contradiction, let us assume that some feasible Nash flow f has bottleneck cost $B(f) > r/(4\gamma_1)$. We recall that f is a Nash flow iff the edges of G' with bottleneck cost $B(f) > r/(4\gamma_1)$ form an $s - t$ cut. This cut does not include the edges of the paths p and q and the external edge e_2, due to the choice of their latencies. Hence, this cut includes a similar cut either in H_6 or in H_9 (or in both), either in H_1 or H_3 (or in both), and either in H_4 or in H_8 (or in H_5 or in H_6, in certain combinations with other subedgeworks, see also Fig. 4.b). Let us consider the case where the edges with bottleneck cost $B(f) > r/(4\gamma_1)$ form a cut in H_1, H_4, and H_6. Namely, the edges of H_1, H_4, and H_6, with bottleneck cost equal to $B(f) > r/(4\gamma_1)$ form an $s - u$, an $s - v$, and an $s - s_2$ cut, respectively, and thus the restriction of f to each of H_1, H_4, and H_6, is an equilibrium flow of bottleneck cost greater than $r/(4\gamma_1)$ for the corresponding routing instance. Since $\mathcal{I}$ is a YES-instance, this can happen only if the flow through H_1 is more than $r/2$, and the flow through each of H_4 and H_6 is more than $r/4$ (see also property (ii) of optimal subedgeworks above). Hence, we obtain that more than r units of flow leave s, a contradiction. All other cases are similar. □

A.5 The Proof of Proposition 6

For convenience, in the proofs of Proposition 6 and Proposition 7, we slightly abuse the terminology, and say that a collection of subedgeworks of H' form an $s - t$ cut, if the union of any cuts in them comprises an $s - t$ cut in H'. Moreover, whenever we write that r_i units of flow are routed through a subedgework H_i, we assume that the routing through H_i corresponds to the worst Nash flow of (H_i, r_i). Also, we recall that since subedgeworks H'_i do not contain any copies of a good subnetwork of D, by property (3) of gap instances, the saturation rate of each H'_i is $r^s_i \leq r/3$, if $i \in \{4, \ldots, 9\}$, and $r^s_i \leq 2r/3$, if $i \in \{1, 3\}$.

We start by showing that either the external edge e_2 is present in H', or $B(H', r) \geq r/(3\gamma_2)$. Indeed, if e_2 is not present in H', the subedgeworks H'_4, H'_6, and H'_8 form an $s-t$ cut in H'. Therefore, we can construct a Nash flow f that routes at least $r/3$ units of flow through H'_4, H'_6, and H'_8, and has $B(f) \geq r/(3\gamma_2)$. Therefore, we can assume, without loss of generality, that e_2 is present in H'.

Similarly, we show that either H' includes at least one path outgoing from s_2 to either t_1 or t_2, and at least one path incoming to t_2 from either s_1 or s_2, or $B(H', r) \geq r/(3\gamma_2)$. In particular, if s_2 is connected to neither t_1 nor t_2, the subedgeworks H'_1 and H'_4 form an $s - t$ cut in H'. Thus, we can construct a Nash flow f that saturates the subedgework H'_1 (or the subedgeworks H'_3 and H'_8, if $r^s_1 > r^s_3 + r^s_8$) and the subedgework H'_4 (or the subedgeworks H'_3 and either H'_5, or H'_9 and at least one of the H'_7 and H'_8, depending on r^s_4 and the saturation rates of the rest). We note that this is always possible with r units of flow, because $r^s_1 \leq 2r/3$ and $r^s_4 \leq r/3$. Therefore, the bottleneck cost of f is $B(f) \geq r/(3\gamma_2)$. In case where there is no path incoming to t_2 from either s_1 or s_2, the subedgeworks H'_3 and H'_8 form an $s - t$ cut in H'. As before, we can construct a Nash flow f that saturates the subedgeworks H'_3 and H'_8 (or, as before, an appropriate combination of other subedgeworks carrying flow to H'_3 and H'_8), and has $B(f) \geq r/(3\gamma_2)$. Therefore, we can assume, without loss of generality, that H' includes at least one path outgoing from s_2 to either t_1 or t_2, and at least one path incoming to t_2 from either s_1 or s_2.

Next, we show that either H' includes at least one path outgoing from s_1 to either t_1 or t_2, and at least one path incoming to t_1 from either s_1 or s_2, or $B(H', r) \geq r/(3\gamma_2)$. In particular, let us consider the case where s_1 is connected to neither t_1 nor t_2 (see also Fig. 5.a, the case where there is no path incoming to t_1 from either s_1 or s_2 can be handled similarly). In the following, we assume
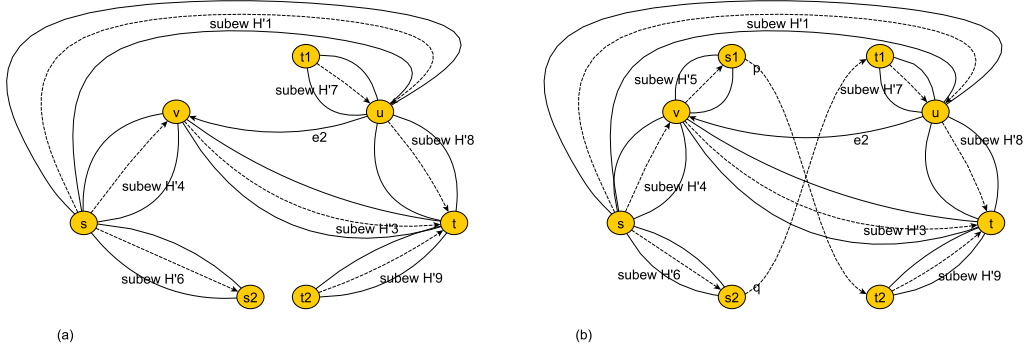
15

Fig. 5. The structure of possible subnetworks of G' when there is no pair of vertex-disjoint paths connecting s_1 to t_1 and s_2 to t_2. The subnetwork (a) contains a path outgoing from s_2 to either t_1 or t_2, and no path outgoing from s_1 to either t_1 or t_2. Hence, no flow can be routed through the edgework G_5, and thus we can regard G_5 as being absent from H'. The subnetwork (b) essentially corresponds to the case where all edges of G' are present in H'.

that s_2 is connected to t_2 (because, by the analysis above, we can assume that there is a path incoming to t_2, and s_1 is not connected to T_2), and construct a Nash flow f of bottleneck cost $B(f) \geq r/(3\gamma_2)$.

We first route $\min\{r_6^s, r_9^s\} \leq r/3$ units of flow through the subedgework H_6', next through an $s_2 - t_2$ path, and finally through the subedgework H_9', and saturate either H_6' or H_9' (or both). If there is an $s_2 - t_1$ path and H_6' is not saturated, we keep routing flow through H_6', next through an $s_2 - t_1$ path, and next through the subedgeworks H_7' and H_8', until either the subedgework H_6' or at least one of the subedgeworks H_7' and H_8' become saturated. Thus, we saturate at least one edgework on every $s - t$ path that includes s_2.

Next, we show how to saturate at least one edgework on every $s - t$ path that includes either v or u. If $r_1^s \leq r_3^s \leq 2r/3$, we route r_1^s units of flow through H_1', e_2, and H_3', and route $\min\{r_3^s - r_1^s, r_4^s\}$ units of flow through H_4' and H_3', and saturate either H_1' and H_3' or H_1' and H_4'. If $r_3^s < r_1^s \leq 2r/3$, we route r_3^s units of flow through H_1', e_2, and H_3', and route $\min\{r_3^s - r_1^s, r_8^s\}$ units of flow through H_1' and H_8', and saturate either H_1' and H_3' or H_3' and H_8'.

The remaining flow (if any) can be routed through these routes, in proportional rates. In all cases, we obtain an $s - t$ cut consisting of saturated subedgeworks. Thus, the resulting flow f is a Nash flow with a bottleneck cost of at least $r/(3\gamma_2)$. $\qquad\square$

A.6 The Proof of Proposition 7

In the following, we consider a subnetwork H' of G' which does not include any copies of a good subnetwork of D, and contains (i) the external edge e_2, (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. Since the copy of the subnetwork of D connecting s_1 and s_2 to t_1 and t_2 in H' is bad, properties (ii) and (iii) imply that H' contains an $s_1 - t_2$ path p and an $s_2 - t_1$ path q. Moreover, since the subedgeworks H_i' do not include any copies of a good subnetwork of D, by property (3) of gap instances, the saturation rate of each H_i' is $r_i^s \leq r/3$, if $i \in \{4, \ldots, 9\}$, and $r_i^s \leq 2r/3$, if $i \in \{1, 3\}$.

We next show that for such a subnetwork H', we can construct a Nash flow f of bottleneck cost $B(f) \geq r/(3\gamma_2)$. At the conceptual level, as in the last case in the proof of Lemma 1, we seek to construct a Nash flow by routing $r/3$ units of flow through each of the following three routes: (i) H_1', e_2, and H_3', (ii) H_1', e_2, H_5', p, and H_9', and (iii) H_6', q, H_7', e_2, and H_3'. However, for simplicity of the analysis, we regard the corresponding (edge) flow as being routed through just two routes: a rate of $2r/3$ is routed through H_1', e_2, and H_3', and a rate of $r/3$ is routed through the (possibly non-simple) route H_6', q, H_7', e_2, H_5', p, and H_9'. We do so because the latter routing allows us to consider fewer

cases in the analysis. We conclude the proof by showing that if the latter route is not simple, we can always decompose the flow into the three simple routes above.

In the following, we assume that with a flow rate of at most $2r/3$, routed through H_1', e_2, and H_3' (and possibly through H_4' and H_8'), we can saturate both subedgeworks H_1' and H_3'. Otherwise, as in the last case in the proof of Proposition 6, we can show how with a total flow rate of at most $2r/3$, part of which is routed through either H_4' or H_8', we can saturate either H_1' and H_4', or H_3' and H_8'. Then, the remaining $r/3$ units of flow can saturate either H_6', in the former case, or H_9', in the latter case. Thus, we obtain a Nash flow with a bottleneck cost of at least $r/(3\gamma_2)$.

Having saturated both subedgeworks H_1' and H_3', using at most $2r/3$ units of flow, we have at least $r/3$ units of flow to saturate the subedgeworks H_5', H_6', H_7', and H_9', or an appropriate subset of them, so that together with H_1' and H_3', they form an $s-t$ cut in H'. We first route $\tau \equiv \min\{r_5^s, r_6^s, r_7^s, r_9^s\} \leq r/3$ units of flow through H_6', q, H_7', e_2, H_5', p, and H_9', until t, and consider different cases, depending on which of the subedgeworks H_5', H_6', H_7', and H_9' has the minimum saturation rate.

- If $\tau = r_9^s$, H_9' is saturated. We first assume that H' contains an $s_1 - t_1$ path, and route (some of) the remaining flow (i) through H_4', H_5', an $s_1 - t_1$ path, H_7', and H_8', and (ii) through H_6', q, H_7', and H_8'. We do so until either at least one of the subedgeworks H_7' and H_8' or the subedgework H_6' and at least one of the subedgeworks H_4' and H_5' become saturated. Since $\min\{r_7^s, r_8^s\} \leq r/3$, this requires at most $r/3-\tau$ additional units of flow. If H' does not contain an s_1-t_1 path, we route the remaining flow only through route (ii), until either at least one of the subedgeworks H_7' and H_8' or the subedgework H_6' become saturated. In both cases, the newly saturated subedgeworks, together with the saturated subedgeworks H_1', H_3', and H_9', form an $s-t$ cut of saturated subedgeworks, and thus the worst equilibrium bottleneck cost is at least $r/(3\gamma_2)$.

- If $\tau = r_6^s$, H_6' is saturated. As before, we first assume that H' contains an $s_1 - t_1$ path, and route the remaining flow (i) through H_4', H_5', p, and H_9', and (ii) through H_4', H_5', an $s_1 - t_1$ path, H_9' and H_8', until either at least one of the subedgeworks H_4' and H_5', or the subedgework H_9' and at least one of the subedgeworks H_7' and H_8' become saturated. Since $\min\{r_4^s, r_5^s\} \leq r/3$, this requires at most $r/3 - \tau$ additional units of flow. If H' does not contain an $s_1 - t_1$ path, we route the remaining flow only through route (i), until either at least one of the subedgeworks H_4' and H_5' or the subedgework H_9' become saturated. In both cases, the newly saturated subedgeworks, together with the saturated subedgeworks H_1', H_3', and H_6', form an $s-t$ cut of saturated subedgeworks, and thus the worst equilibrium bottleneck cost is at least $r/(3\gamma_2)$.

- If $\tau = r_7^s$, H_7' is saturated. Then, we first assume that H' contains an $s_2 - t_2$ path, and route the remaining flow (i) through H_4', H_5', p, and H_9', and (ii) through H_6', an $s_2 - t_2$ path, and H_9', until either the subedgework H_9', or the subedgework H_6' and at least one of the subedgeworks H_4' and H_5' become saturated. Since $r_9^s \leq r/3$, this requires at most $r/3 - \tau$ additional units of flow. If H' does not contain an $s_2 - t_2$ path, we route the remaining flow only through route (i), until either at least one of the subedgeworks H_4' and H_5' or the subedgework H_9' become saturated. In both cases, the newly saturated subedgeworks, together with the saturated subedgeworks H_1', H_3', and H_7', form an $s-t$ cut of saturated subedgeworks, and thus the worst equilibrium bottleneck cost is at least $r/(3\gamma_2)$.

- If $\tau = r_5^s$, H_5' is saturated. As before, we first assume that H' contains an $s_2 - t_2$ path, and route the remaining flow (i) through H_6', q, H_7', and H_8', and (ii) through H_6', an $s_2 - t_2$ path, and H_9', until either the subedgework H_6', or the subedgework H_9' and at least one of the subedgeworks H_7' and H_8' become saturated. Since $r_6^s \leq r/3$, this requires at most $r/3 - \tau$ additional units of flow. If H' does not contain an $s_2 - t_2$ path, we route the remaining flow only through route (i), until either at least one of the subedgeworks H_7' and H_8' or the subedgework H_6' become saturated. In

both cases, the newly saturated subedgeworks, together with the saturated subedgeworks H_1', H_3', and H_5', form an $s - t$ cut of saturated subedgeworks, and thus the worst equilibrium bottleneck cost is at least $r/(3\gamma_2)$.

Thus, in all cases, we obtain an equilibrium flow with a bottleneck cost of at least $r/(3\gamma_2)$. However, in the construction above, the route H_6', q, H_7', e_2, H_5', p, H_9' may not be simple, since p and q may not be vertex-disjoint. If this is the case, this route is technically not allowed by our model, where the flow is only routed through simple $s - t$ paths. Nevertheless, the corresponding edge flow can be decomposed into the following three simple routes: (i) H_1', e_2, and H_3', (ii) H_1', e_2, H_5', p, and H_9', and (iii) H_6', q, H_7', e_2, and H_3', unless $\min\{r_1^s, r_3^s\} \le r/3$. Moreover, if $\min\{r_1^s, r_3^s\} \le r/3$, we can work as above, and saturate both H_1' and H_3' with at most $r/3$ units of flow. The remaining $2r/3$ units of flow can be routed (i) through H_6', q, H_7', and H_8', and (ii) through H_4', H_5', p, and H_9', and possibly either through H_6', an $s_2 - t_2$ path[6], and H_9', or through H_4', H_5', an $s_1 - t_1$ path, H_7', and H_8', until either H_4' (or H_5') and H_6', or H_7' (or H_8') and H_9' are saturated. This routing only uses simple routes. In addition, these saturated subedgeworks, together with the saturated subedgeworks H_1' and H_3', form an $s - t$ cut of saturated subedgeworks, and thus the worst equilibrium bottleneck cost is at least $r/(3\gamma_2)$. □

A.7 Theorem 3: The Proofs of (ii), (iii), and (iv)

We first show (ii), i.e., that $B(f) \le B(H^*, 1) + \varepsilon$. We recall that H^* denotes the best subnetwork of $\mathcal{G}$ and f^* denotes the worst Nash flow in H^*. Also, by hypothesis, $f_e^* > \delta > 0$, for all edges e of H^*.

By Lemma 3, there is a candidate flow $\tilde{f}$ such that for all edges e of H^*, $|\tilde{f}_e - f_e^*| \le \epsilon_1$. Thus, since $\epsilon_1 \le \delta$, H^* is a candidate network, because $\tilde{f}_e > 0$ for all edges e of H^*. Moreover, by the Lipschitz condition and the choice of ϵ_1, for all edges e of H^*, $|c_e(\tilde{f}_e) - c_e(f_e^*)| \le \varepsilon/4$. Therefore, since f^* is a Nash flow in H^*, $\tilde{f}$ is an ϵ_2-Nash flow in H^*, and thus $(H, \tilde{f})$ is a candidate solution. Furthermore, $|B(\tilde{f}) - B(f^*)| \le \varepsilon/4$, i.e., the bottleneck cost of $\tilde{f}$ is within an additive term of $\varepsilon/4$ from the worst equilibrium bottleneck cost of H^*. In particular, $B(\tilde{f}) \le B(H^*, 1) + \varepsilon/4$.

We also need to show that for any other candidate flow g for which (H^*, g) is a candidate solution, $B(g) \le B(\tilde{f}) + 3\varepsilon/4$, and thus $\tilde{B}(H^*) \le B(\tilde{f}) + 3\varepsilon/4 \le B(H^*, 1) + \varepsilon$. To reach a contradiction, let us assume that there is a candidate flow g that is an ϵ_2-Nash flow in H^* and has $B(g) > B(\tilde{f}) + 3\varepsilon/4$. But then, we should expect that there is a Nash flow g' in H^* that closely approximates g and has a bottleneck cost of $B(g') \approx B(g) > B(f^*)$, a contradiction. Formally, since g is an ϵ_2-Nash flow in H^*, the set of edges with $c_e(g_e) \ge B(g) - \varepsilon/2$ comprises an $s - t$ cut in H^*. Then, by the continuity of the latency functions, we can fix a part of the flow routed essentially as in g, so that there is an $s - t$ cut consisting of used edges with latency $B(g) - \varepsilon/2$, and possibly unused edges with latency at least $B(g) - \varepsilon/2$, and reroute the remaining flow on top of it, so that we obtain a Nash flow g' in H^*. But then,
$$B(g') \ge B(g) - \varepsilon/2 > B(\tilde{f}) + \varepsilon/4 \ge B(f^*)\,,$$
which contradicts the hypothesis that f^* is the worst Nash flow in H^*.

Therefore, $\tilde{B}(H^*) \le B(H^*, 1) + \varepsilon$. Since the algorithm returns the candidate solution (H, f), and not a candidate solution including H^*, $\tilde{B}(H) \le \tilde{B}(H^*)$. Thus, we obtain (ii), namely that $\tilde{B}(H) = B(f) \le B(H^*, 1) + \varepsilon$.

We proceed to show (iii), namely that $B(H, 1) \le B(f) + \varepsilon/4$. To this end, we let g be the worst Nash flow in H. By Lemma 3, there is a candidate flow $\tilde{g}$ such that for all edges e of H, $|\tilde{g}_e - g_e| \le \epsilon_1$,

---

[6] We note that if the paths p and q are not vertex-disjoint, we also have an $s_1 - t_1$ path and an $s_2 - t_2$ path in H'.

if $g_e > 0$, and $\tilde{g}_e = 0$, otherwise. Therefore, by the Lipschitz condition and the choice of ϵ_1, for all edges e of H, $|c_e(\tilde{g}_e) - c_e(g_e)| \leq \varepsilon/4$, if $g_e > 0$, and $c_e(\tilde{g}_e) = c_e(g_e) = 0$, otherwise. This implies that $|B(\tilde{g}) - B(g)| \leq \varepsilon/4$, i.e., that bottleneck cost of $\tilde{g}$ is within an additive term of $\varepsilon/4$ from the bottleneck cost of g. In particular, $B(g) \leq B(\tilde{g}) + \varepsilon/4$.

We also need to show that $(H, \tilde{g})$ is a candidate solution. Since H is a candidate subnetwork and $\tilde{g}$ is a candidate flow, we only need to show that $\tilde{g}$ is an ϵ_2-Nash flow in H. Since g is a Nash flow in H, the set of edges $C = \{e : c_e(g_e) \geq B(g)\}$ comprises an $s - t$ cut in H. In fact, for all edges $e \in C$, $c_e(g_e) = B(g)$, if $g_e > 0$, and $c_e(g_e) \geq B(g)$, otherwise. Let us now consider the latency in $\tilde{g}$ of each edge $e \in C$. If $g_e = 0$, then $c_e(\tilde{g}_e) = c_e(g_e) \geq B(g) \geq B(\tilde{g}) - \varepsilon/4$. If $g_e > 0$, then

$$B(\tilde{g}) \geq c_e(\tilde{g}_e) \geq c_e(g_e) - \varepsilon/4 = B(g) - \varepsilon/4 \geq B(\tilde{g}) - \varepsilon/2\,.$$

Therefore, for the flow $\tilde{g}$, we have an $s - t$ cut in H consisting of edges e either with $\tilde{g}_e > 0$ and $B(\tilde{g}) - \varepsilon/2 \leq c_e(\tilde{g}_e) \leq B(\tilde{g})$, or with $\tilde{g}_e = 0$ and $c_e(\tilde{g}_e) \geq B(\tilde{g}) - \varepsilon/4$. By the standard properties of ε-Nash flows (see also in Section 2), we obtain that $\tilde{g}$ is a ϵ_2-Nash flow in H.

Hence, we have shown that $(H, \tilde{g})$ is a candidate solution, and that $B(g) \leq B(\tilde{g}) + \varepsilon/4$. Therefore, the algorithm considers both candidate solutions (H, f) and $(H, \tilde{g})$, and returns (H, f), which implies that $B(\tilde{g}) \leq B(f)$. Thus, we obtain (iii), namely that $B(H, 1) = B(g) \leq B(f) + \varepsilon/4$.

To conclude the proof, we next show (iv), namely that $B(f) \leq B(H, 1) + \varepsilon/2$. For the proof, we use the same notation as in (iii). The argument is essentially identical to that used in the second part of the proof of (ii). More specifically, to reach a contradiction, we assume that the candidate flow f, which is an ϵ_2-Nash flow in H, has $B(f) > B(H, 1) + \varepsilon/2$. Then, as before, we should expect that there is a Nash flow f' in H that approximates f and has a bottleneck cost of $B(f') \approx B(f) > B(H, 1)$, a contradiction. Formally, since f is an ϵ_2-Nash flow in H, the set of edges with $c_e(f_e) \geq B(f) - \varepsilon/2$ comprises an $s - t$ cut in H. Then, by the continuity of the latency functions, we can fix a part of the flow routed essentially as in f, so that there is an $s - t$ cut consisting of used edges with latency $B(f) - \varepsilon/2$, and possibly unused edges with latency at least $B(f) - \varepsilon/2$, and reroute the remaining flow on top of it, so that we obtain a Nash flow f' in H. But then, $B(f') \geq B(f) - \varepsilon/2 > B(H, 1)$, which contradicts the definition of the worst equilibrium bottleneck cost $B(H, 1)$ of H. Thus, we obtain (iv), namely that $B(f) \leq B(H, 1) + \varepsilon/2$. □