# On the Severity of Braess's Paradox: Designing Networks for Selfish Users Is Hard*

Tim Roughgarden†

January 20, 2004

**Abstract**

We consider a directed network in which every edge possesses a latency function that specifies the time needed to traverse the edge given its congestion. Selfish, noncooperative agents constitute the network traffic and wish to travel from a source vertex $s$ to a destination $t$ as quickly as possible. Since the route chosen by one network user affects the congestion experienced by others, we model the problem as a noncooperative game. Assuming that each agent controls only a negligible portion of the overall traffic, Nash equilibria in this noncooperative game correspond to $s$-$t$ flows in which all flow paths have equal latency.

A natural measure for the performance of a network used by selfish agents is the common latency experienced by users in a Nash equilibrium. Braess's Paradox is the counterintuitive but well-known fact that removing edges from a network can *improve* its performance. Braess's Paradox motivates the following network design problem: given a network, which edges should be removed to obtain the best flow at Nash equilibrium? Equivalently, given a network of edges that can be built, which subnetwork will exhibit the best performance when used selfishly?

We give optimal inapproximability results and approximation algorithms for this network design problem. For example, we prove that there is no approximation algorithm for this problem with approximation ratio less than $n/2$, where $n$ is the number of network vertices, unless $P = NP$. We further show that this hardness result is the best possible, by exhibiting an $(n/2)$-approximation algorithm. We also prove tight inapproximability results when additional structure, such as linearity, is imposed on the network latency functions.

Moreover, we prove that an optimal approximation algorithm for these problems is the *trivial algorithm*: given a network of candidate edges, build the entire network. As a consequence, we show that Braess's Paradox—even in its worst-possible manifestations—is impossible to detect efficiently.

*A preliminary version of this paper appeared in the Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, October 2001.

†Computer Science Division, UC Berkeley, 595 Soda Hall, Berkeley, CA 94720. Supported by an NSF Postdoctoral Fellowship. This research was done while the author was at Cornell University and supported by an NSF Graduate Fellowship, a Cornell University Fellowship, and ONR grant N00014-98-1-0589. Email: timr@cs.berkeley.edu.

En route to these results, we give a fundamental generalization of Braess's Paradox: the improvement in performance that can be effected by removing edges can be arbitrarily large in large networks. Even though Braess's Paradox has enjoyed 35 years as a textbook example, our result is the first to extend its severity beyond that in Braess's original four-node network.

# 1 Introduction

### Selfish Routing

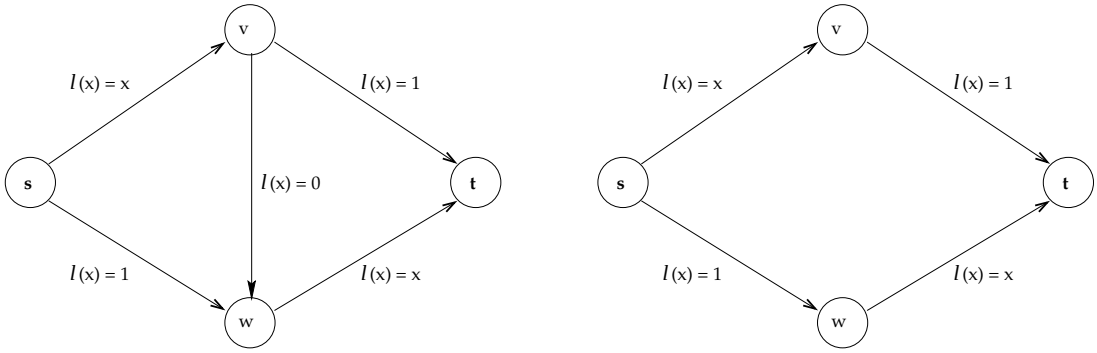
A central and well-studied problem arising in the management of a large network is that of routing traffic to achieve the best possible network performance. Recently, researchers have started to confront the harsh reality that in many networks, it is difficult or even impossible to impose optimal or near-optimal routing strategies on network traffic, leaving network users free to act according to their own interests. This realization has led many researchers, too numerous to list here, to employ classical game theory and model the behavior of network users with *noncooperative games* and their *Nash equilibria*. For a gentle introduction to basic game-theoretic concepts, see Straffin [102].

Nash equilibria can be *inefficient*, in the sense that they need not optimize natural global objective functions [38]. This fact has motivated researchers to propose several different ways of *coping with selfishness*—of ensuring that selfish behavior results in a socially desirable outcome. The previous approaches explored by the theoretical computer science community for traffic routing problems include bounding the worst-possible inefficiency of Nash equilibria, also known as the "price of anarchy" [20, 29, 31, 32, 68, 75, 81, 90, 91, 93, 94, 107], influencing the behavior of selfish agents via pricing policies [1, 21, 26, 27] and network switch protocols [98], and routing a small portion of the traffic centrally [64, 88].

In this paper, we explore a different idea for ameliorating the degradation in network performance due to selfish routing: armed with the knowledge that our networks will be host to selfish users, how can we design them to minimize the inefficiency inherent in a Nash equilibrium?

### Braess's Paradox

We will consider a directed network in which each edge possesses a latency function specifying the time needed to traverse the edge given its congestion, and will assume that all network traffic wishes to travel from a distinguished source vertex $s$ to a destination vertex $t$. Selfish, noncooperative agents constitute the network traffic, and each wishes to travel from $s$ to $t$ as quickly as possible. Since the route chosen by one network user affects the congestion, and hence the latency, experienced by others, it will be useful to view the problem as a noncooperative game. Assuming each agent controls only a negligible portion of the overall traffic, an assignment of traffic to paths in the network can be modeled as fractional network flow, with a Nash equilibrium in the noncooperative game corresponding to an $s$-$t$ flow in which all flow paths have equal—and minimum-possible—latency. Informal discussion of this model began in the 1920's [62, 85], and the model was rigorously defined in the 1950's [13, 109]. Many more references can be found in [43, 78, 92, 97].




(a) The Whole Graph

(b) The Optimal Subgraph

Figure 1: Braess's Paradox. Removing an edge from a network can improve its performance.

A natural measure for the performance of a network used by selfish agents is the common latency experienced by each user in a Nash equilibrium, as it navigates from $s$ to $t$. *Braess's Paradox* is the counterintuitive but well-known fact that removing edges from a network can *improve* its performance. We illustrate this phenomenon with a simple example, shown in Figure 1, that is due to Schulman [95] and closely based on Braess's original example [15]. In the figure, every edge is labeled with its latency function, which is a function of the amount of flow $x$ that uses the edge. For example, if there is one half unit of flow on the edge $(s, v)$, then all of this flow experiences one half unit of latency when traversing the edge. Now suppose that one unit of flow needs to be routed from $s$ to $t$ in the network in Figure 1(a). In the unique flow at Nash equilibrium, all traffic follows the path $s \to v \to w \to t$ and experiences two units of latency; since the other two $s$-$t$ paths also have latency 2 with respect to this flow, no user has an incentive to switch paths. On the other hand, suppose we remove the edge $(v, w)$, thereby obtaining the network in Figure 1(b). Then, in the unique flow at Nash equilibrium, half of the flow travels on each of the two paths, and all agents experience $\frac{3}{2}$ units of latency.

### Network Design

Braess's Paradox immediately suggests the following network design problem: given a network with latency functions on the edges and a traffic rate, which edges should be removed to obtain the best possible flow at Nash equilibrium? Equivalently, given a large network of candidate edges to build, which subnetwork will exhibit the best performance when used selfishly?

This problem is fundamentally different from most network design problems that have been studied by the theoretical computer science community, such as those described in [50, 61, 86], which typically ask for the smallest or cheapest network satisfying desiderata such as high connectivity or small diameter. Problems of this sort are only non-trivial in the presence of non-zero costs on vertices and/or edges—otherwise, the best solution is simply to build

the largest possible network. Braess's Paradox shows that this approach is suboptimal for our network design problem: there are no costs, yet it is not at all clear which network should be preferred.

Ever since Braess's Paradox was reported [15, 77], researchers have attempted to solve variants of this network design problem (e.g., [34]), but scant progress has been made either computationally or theoretically. Indeed, early computational work either focused on very small networks [69] or admitted to ignoring congestion effects entirely, due to the difficulties involved [14, 36, 54, 87, 96, 112]. In a 1984 survey, Magnanti and Wong describe the problem as "essentially unsolved" from a practical perspective [73, P.15].

On the theoretical side, all existing work on the network design problem that we study and on the problem of detecting Braess's Paradox either exclusively considers the four-node network of Figure 1(a) [45, 56, 82, 83] or other very special classes of networks [46], or focuses entirely on the special case where only *one* edge is to be added or deleted from the network [33, 101, 103]. Since the latency of traffic at Nash equilibrium can be computed in polynomial time (see [92]), restricting attention to single edge removals yields an algorithmically trivial problem, which can be solved in polynomial time by enumerating all possible solutions. In our network design problem, any subset of edges can be removed, and there can thus be exponentially many candidate solutions. The network design problem also becomes easier if the goal is to determine whether a graph, unadorned with latency functions, is vulnerable to Braess's Paradox in the sense that *some* assignment of latency functions to edges causes Braess's Paradox to arise [76, 77]. In our version of the problem, a network is given with latency functions already attached. Finally, a recent series of papers [4, 39, 65, 66, 70] studies a similar network design problem in capacitated networks, of how to allocate a fixed amount of capacity to edges to obtain the largest possible improvement in the Nash equilibrium. These papers either confine themselves to networks of parallel links or provide only sufficient, and far from necessary, conditions for a given capacity allocation to improve network performance. They are therefore not directly relevant for our network design problem.

Very recently, the network design problem described in this paper has received attention from the theoretical computer science community, and several researchers have asked if there are efficient exact or near-optimal algorithms for the problem.

Designing networks for selfish users has thus appeared difficult from a range of perspectives and to several research communities. In this paper, we present a theoretical explanation for this perceived difficulty.

### Our Results: Network Design

We give optimal inapproximability results and approximation algorithms for several network design problems of the following type: given a network with edge latency functions, a single source-sink pair, and a rate of traffic, find the subnetwork minimizing the latency of all (selfish) network users in a flow at Nash equilibrium. Specifically, we prove the following for every $\epsilon > 0$, assuming $P \neq NP$:

- General Latency Network Design: For networks with continuous, non-negative, non-decreasing edge latency functions, there is no $(\lfloor n/2 \rfloor - \epsilon)$-approximation algorithm[1] for network design, where $n$ is the number of vertices in the network. We also prove that this hardness result is the best possible by exhibiting an $\lfloor n/2 \rfloor$-approximation algorithm for the problem.

- Linear Latency Network Design: For networks in which the latency of each edge is a linear function of the congestion, there is no $(\frac{4}{3} - \epsilon)$-approximation algorithm for network design. The existence of a $\frac{4}{3}$-approximation algorithm follows easily from existing work, which proves that this hardness result is optimal.

These problems were not previously known to be NP-hard, and no finite approximation ratio for General Latency Network Design was previously known. Our hardness proofs do not require the machinery of probabilistically checkable proofs (see, e.g., [8, 9, 41]), and are direct "gap" reductions from well-known NP-complete problems.

Moreover, we prove that an optimal approximation algorithm for these problems is what we call the *trivial algorithm*: given a network of candidate edges, build the entire network. As a consequence of the optimality of the trivial algorithm, we prove that Braess's Paradox—the presence of harmful extraneous edges—is impossible to detect efficiently, even in its worst-possible manifestations.

Finally, we show that our strong hardness results are not particular to the classes of general and linear latency functions. Rather, for additional classes of latency functions, such as polynomials of bounded degree, the trivial algorithm achieves the best possible performance guarantee (up to a constant factor).

### Our Results: Braess's Paradox

Braess's Paradox has intrigued researchers ever since its discovery 35 years ago [15], appearing frequently in textbooks [30, 48, 67, 79, 80, 97] and the popular science literature [7, 10, 11, 22, 63, 84]. In addition, Braess's Paradox has spawned many further research developments. It has catalyzed the search for other "paradoxes" in traffic networks [6, 19, 35, 42, 52, 57, 58, 60, 99, 100, 108, 113] and for analogues of Braess's Paradox in queueing networks [18, 24, 25, 111] as well as in seemingly unrelated contexts [12, 17, 23, 59]. It has renewed interest in the older "Downs-Thomson paradox" [16, 37, 106]. It has even stirred debate over its implications for classical philosophical problems [55, 74].

Despite this storied history, it was not previously known whether Braess's Paradox could be more severe than it is in Braess's original network [15, 95]. En route to proving our strongest inapproximability result, we resolve this open question in the affirmative. In fact, we do so in a very strong way: there in an infinite family of networks that demonstrates that Braess's Paradox can be arbitrarily severe. Moreover, we prove that our construction is in some sense the best possible.

[1] A *$c$-approximation algorithm* for a minimization problem runs in polynomial time and returns a solution no more than $c$ times as costly as an optimal solution. The value $c$ is the *approximation ratio* or *performance guarantee* of the algorithm.

**Organization**

In Section 2 we formally define our model, review several of its important properties, and prove some preliminary results needed in subsequent sections. In Sections 3-5, we prove matching upper and lower bounds on the best achievable approximation ratio for network design for several classes of edge latency functions. Linear latency functions are considered in Section 3. We consider networks with arbitrary latency functions in Section 4. We also give our construction of an arbitrarily severe Braess's Paradox in this section. In Section 5 we extend our inapproximability results to other classes of latency functions, including degree-bounded polynomials.

# 2 Preliminaries

## 2.1 The Model

We consider a directed network $G = (V, E)$ with vertex set $V$, edge set $E$, and a distinguished source vertex $s$ and sink vertex $t$. We allow multiple edges between vertices but have no use for self-loops. We denote the set of (simple) $s$-$t$ paths by $\mathcal{P}$, which we assume is non-empty. A *flow* is a function $f : \mathcal{P} \to \mathcal{R}^+$. A flow induces a *flow on edges* $\{f_e\}_{e \in E}$ with $f_e = \sum_{P:e \in P} f_P$ the amount of flow on edge $e$. While a flow induces a unique flow on edges, a flow on edges can correspond to many different flows (on paths). With respect to a finite and positive *traffic rate* $r$, a flow $f$ is said to be *feasible* if $\sum_{P \in \mathcal{P}} f_P = r$. Each edge $e \in E$ possesses a congestion-dependent *latency* that we denote by $\ell_e(\cdot)$. The latency of a path $P$ with respect to a flow $f$ is then the sum of the latencies of the edges in the path, denoted by $\ell_P(f) = \sum_{e \in P} \ell_e(f_e)$. For each edge $e \in E$, we assume that the latency function $\ell_e$ is non-negative, continuous, and non-decreasing. We call the triple $(G, r, \ell)$ an *instance*.

## 2.2 Flows at Nash Equilibrium

We will consider flows that represent an equilibrium among many noncooperative agents—flows that behave in a "greedy" or "selfish" manner. Intuitively, we expect each unit of such a flow, no matter how small, to travel along the minimum-latency path available to it, where latency is measured with respect to the rest of the flow. If this condition failed, then this flow would reroute itself on a path with smaller latency. Following [34, 93], we formalize this idea in the next definition.

**Definition 2.1** A flow $f$ feasible for $(G, r, \ell)$ is at *Nash equilibrium*, or is a *Nash flow*, if for all $P_1, P_2 \in \mathcal{P}$ with $f_{P_1} > 0$ and $\delta \in (0, f_{P_1}]$, we have

$$\ell_{P_1}(f) \leq \ell_{P_2}(\tilde{f}),$$

where

$$\tilde{f}_P = \begin{cases} f_P - \delta & \text{if } P = P_1 \\ f_P + \delta & \text{if } P = P_2 \\ f_P & \text{otherwise.} \end{cases}$$

Letting $\delta$ tend to 0, continuity and monotonicity of the edge latency functions give the following useful characterization of a flow at Nash equilibrium, first stated by Wardrop [109].

**Proposition 2.2** *A flow $f$ feasible for $(G, r, \ell)$ is at Nash equilibrium if and only if for every $P_1, P_2 \in \mathcal{P}$ with $f_{P_1} > 0$,*

$$\ell_{P_1}(f) \leq \ell_{P_2}(f).$$

In particular, if $f$ is at Nash equilibrium then all of the $s$-$t$ paths to which $f$ assigns a positive amount of flow have equal latency.

The following proposition states that, under our assumption of continuous, non-decreasing edge latency functions, flows at Nash equilibrium always exist and are essentially unique.

**Proposition 2.3** *Every instance $(G, r, \ell)$ admits a flow at Nash equilibrium. Moreover, if $f, f'$ are Nash flows for $(G, r, \ell)$, then $\ell_P(f) = \ell_P(f')$ for every $s$-$t$ path $P$.*

Proposition 2.3 is due to Beckmann, McGuire, and Winsten [13], and proofs can also be found in [15, 34, 93]. These proofs of Proposition 2.3 show that the Nash flows of an instance are precisely the optimal solutions of a particular (and peculiar) convex program. The objective function that is minimized in this convex program is a non-decreasing function of the induced flow on edges, and is independent of the particular flow decomposition. Every Nash flow can therefore be made acyclic, by removing flow cycles from its induced flow on edges and then taking an arbitrary path decomposition.

**Proposition 2.4** *Every instance $(G, r, \ell)$ admits a flow $f$ at Nash equilibrium with the property that the subgraph of $G$ of edges for which $f_e > 0$ is directed acyclic.*

## 2.3 Formalizing the Network Design Problem

Let $L(G, r, \ell)$ be the common latency of every $s$-$t$ flow path of a flow $f$ at Nash equilibrium for $(G, r, \ell)$. The value $L(G, r, \ell)$ is well defined by Propositions 2.2 and 2.3. When no confusion results, we will abbreviate the expression $L(G, r, \ell)$ by $L(G)$.

We may thus formally state our network design problem as follows:

Given an instance $(G, r, \ell)$, find a subgraph $H$ of $G$ that minimizes $L(H, r, \ell)$.

Since the value $L(H, r, \ell)$ can be recovered from the subgraph $H$ in polynomial time via convex programming (see [92]), we can view this problem as purely combinatorial.

**Remark 2.5** Strictly speaking, to talk about "polynomial-time algorithms", we need to prescribe how instances are encoded for input to a (mathematical model of a) computer. This is not a trivial problem, since in our exposition we are permitting network latency functions to be arbitrary continuous, non-decreasing functions. Nonetheless, these and similar issues have been thoroughly treated elsewhere, such as in [2, 3, 49, 51], and we will not bore the reader with a detailed discussion of this point. Suffice it to say that our results hold with any reasonable encoding method. For example, all of our hardness results hold even if latency functions are restricted to be piecewise linear with a constant number of breakpoints.

## 2.4 The Cost of a Flow

Our next preliminary result relates our objective function $L(H, r, \ell)$ to a second objective function that has been well studied. This connection will be crucial for proving upper bounds on the performance guarantee of the trivial algorithm in Sections 3 and 5.

**Definition 2.6** The *cost* $C(f)$ of a flow $f$ feasible for $(G, r, \ell)$ is the total latency incurred by $f$:

$$C(f) = \sum_{P \in \mathcal{P}} \ell_P(f) f_P.$$

We immediately see that the cost of a flow at Nash equilibrium can be written in a particularly nice form.

**Proposition 2.7** *If $f$ is a flow at Nash equilibrium for $(G, r, \ell)$, then*

$$C(f) = r \cdot L(G, r, \ell).$$

## 2.5 Properties of Nash Flows

In this subsection we note two useful properties of Nash flows. The first states that $L(G, r, \ell)$ is non-decreasing in the traffic rate $r$, when the graph $G$ and latency functions $\ell$ are fixed. We will use this result in Subsection 5.1.

**Proposition 2.8** *For every instance $(G, r, \ell)$, $L(G, r, \ell)$ is a non-decreasing function of $r$.*

Proposition 2.8 was first established by Hall [53]. For a combinatorial proof, see Lin, Roughgarden, and Tardos [72].

Next, we will show that a certain type of vertex ordering exists relative to a directed acyclic Nash flow. Such orderings will be an important tool in Subsection 4.1.

**Definition 2.9** Let $f$ be a Nash flow for the instance $(G, r, \ell)$. Let $d(v)$ denote the length of a shortest $s$-$v$ path with respect to the edge lengths $\{\ell_e(f_e)\}_{e \in E}$. An ordering of the vertices of $G$ is *$f$-monotone* if it satisfies the following two properties.

(P1) All $f$-flow travels forward in the ordering.

(P2) The $d$-values of vertices are non-decreasing in the ordering.

We will show that properties (P1) and (P2) are compatible, in the sense that an $f$-monotone ordering exists relative to a directed acyclic Nash flow. To prove this, we require the following alternative definition of Nash flows.

**Proposition 2.10** *Let $f$ be a flow feasible for $(G, r, \ell)$. For a vertex $v$ in $G$, let $d(v)$ denote the length, with respect to edge lengths $\{\ell_e(f_e)\}_{e \in E}$, of a shortest $s$-$v$ path in $G$. Then $f$ is at Nash equilibrium if and only if*

$$d(w) - d(v) \leq \ell_e(f_e) \tag{1}$$

*for all edges $e = (v, w)$, with equality holding whenever $f_e > 0$.*

*Proof:* First, we note that (1)—the "triangle inequality"—holds for every flow $f$ by the definition of the shortest path labels $d$. Thus, for an $s$-$t$ path $P \in \mathcal{P}$, we can write

$$\begin{aligned} \ell_P(f) &= \sum_{e \in P} \ell_e(f_e) \\ &\geq \sum_{e=(v,w) \in P} d(w) - d(v) \qquad (2) \\ &= d(t) - d(s) = d(t). \end{aligned}$$

By the definition of $d(t)$, a path $P$ is minimum-latency with respect to $f$ if and only if $\ell_P(f) = d(t)$. This occurs if and only if equality holds in (2), which in turn is true if and only if $d(w) - d(v) = \ell_e(f_e)$ for all edges $e$ of $P$. The proposition now follows from Proposition 2.2. ■

We can now prove that an $f$-monotone ordering exists relative to a directed acyclic Nash flow $f$.

**Proposition 2.11** *If $f$ is a directed acyclic Nash flow for the instance $(G, r, \ell)$, then there is an $f$-monotone ordering of the vertices of $G$.*

*Proof:* First, topologically sort the vertices of $G$ according to the directed acyclic flow $f$ to ensure property (P1) of Definition 2.9. Define distance labels $d$ as in Definition 2.9. An ordered pair $(u, v)$ of vertices is *bad* if $d(v) < d(u)$ in spite of $v$ following $u$ in the ordering. Property (P2) of Definition 2.9 is equivalent to the absence of bad vertex pairs.

If there is a bad vertex pair, there is one such pair $(u, v)$ with $u$ and $v$ adjacent in the ordering. By Proposition 2.10 and the non-negativity of latency functions, $d$-values cannot decrease across an edge $e$ with $f_e > 0$. Hence, there is no flow-carrying edge from $u$ to $v$. Transposing $u$ and $v$ therefore does not violate property (P1) of Definition 2.9 and strictly decreases the number of bad vertex pairs. Finitely many such transpositions must therefore yield an $f$-monotone ordering. ■

# 3 Linear Latency Functions

We begin with networks in which the latency of every edge is a linear function of the congestion, so that each latency function $\ell_e$ can be written $\ell_e(x) = a_e x + b_e$ for $a_e, b_e \geq 0$. This is a commonly studied scenario [34, 45, 46, 47, 82, 83, 93, 100, 101], and our proof of inapproximability is particularly simple in this case.

Recall that the *trivial algorithm*, when presented with instance $(G, r, \ell)$, outputs the entire network $G$. We begin by observing that the trivial algorithm is a $\frac{4}{3}$-approximation algorithm for Linear Latency Network Design. This will follow easily from a result of Roughgarden and Tardos [93]. The result states that in every network with linear latency functions, the cost of a flow at Nash equilibrium is at most 4/3 times that of every other feasible flow.

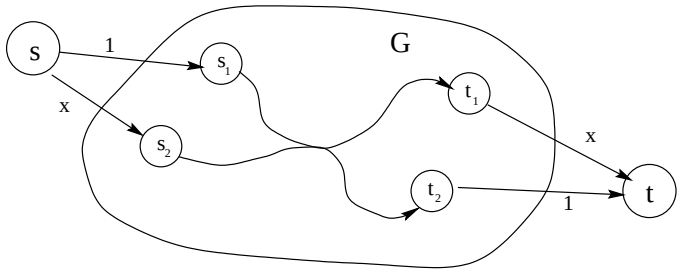



Figure 2: Proof of Theorem 3.3. In a "no" instance of 2DDP, the existence of $s_1$-$t_1$ and $s_2$-$t_2$ paths implies the existence of an $s_2$-$t_1$ path.

**Proposition 3.1 ([93])** *Let $f^*$ and $f$ be feasible and Nash flows, respectively, for an instance $(G, r, \ell)$ with linear latency functions. Then,*

$$C(f) \leq \frac{4}{3} \cdot C(f^*).$$

**Corollary 3.2** *The trivial algorithm is a $\frac{4}{3}$-approximation algorithm for* Linear Latency Network Design.

*Proof:* Consider an instance $(G, r, \ell)$ with linear latency functions, with subgraph $H$ minimizing $L(H, r, \ell)$. Let $f$ and $f^*$ denote flows at Nash equilibrium for $(G, r, \ell)$ and $(H, r, \ell)$, respectively. By Proposition 2.7, we can write $C(f) = r \cdot L(G, r, \ell)$ and $C(f^*) = r \cdot L(H, r, \ell)$. Since $f^*$ can be viewed as a feasible flow for $(G, r, \ell)$, Proposition 3.1 implies that $C(f) \leq \frac{4}{3} C(f^*)$ and hence $L(G, r, \ell) \leq \frac{4}{3} L(H, r, \ell)$. ■

The main result of this section is that, unless $P = NP$, no better approximation is possible in polynomial time.

**Theorem 3.3** *Assuming $P \neq NP$, for every $\epsilon > 0$ there is no $(\frac{4}{3} - \epsilon)$-approximation algorithm for* Linear Latency Network Design.

*Proof:* Our reduction will be from the problem 2 Directed Disjoint Paths (2DDP): given a directed graph $G = (V, E)$ and distinct vertices $s_1, s_2, t_1, t_2 \in V$, are there $s_i$-$t_i$ paths $P_i$ for $i = 1, 2$, such that $P_1$ and $P_2$ are vertex-disjoint? Fortune, Hopcroft, and Wyllie [44] proved that this problem is NP-complete. We will show that a $(\frac{4}{3} - \epsilon)$-approximation algorithm for Linear Latency Network Design can be used to distinguish "yes" and "no" instances of 2DDP in polynomial time.

Consider an instance $\mathcal{I}$ of 2DDP, as above. Augment the vertex set $V$ by an additional source $s$ and sink $t$, and include directed edges $(s, s_1)$, $(s, s_2)$, $(t_1, t)$, and $(t_2, t)$, see Figure 2. Denote the new network by $G' = (V', E')$ and endow the edges of $E'$ with the following linear latency functions $\ell$: edges of $E$ are given the latency function $\ell(x) = 0$, edges $(s, s_2)$ and $(t_1, t)$ are given the latency function $\ell(x) = x$, and edges $(s, s_1)$ and $(t_2, t)$ are given the latency function $\ell(x) = 1$. The instance $(G', 1, \ell)$ can be constructed from $\mathcal{I}$ in polynomial time.

To complete the proof, it suffices to show the following two statements.

(i) If $\mathcal{I}$ is a "yes" instance of 2DDP, then $G'$ admits a subgraph $H$ of $G'$ with $L(H, 1, \ell) = \frac{3}{2}$.

(ii) If $\mathcal{I}$ is a "no" instance, then $L(H, 1, \ell) \geq 2$ for all subgraphs $H$ of $G'$.

To prove (i), let $P_1$ and $P_2$ be vertex-disjoint $s_1$-$t_1$ and $s_2$-$t_2$ paths in $G$, respectively, and obtain $H$ by deleting all edges of $G$ not contained in some $P_i$. Then, $H$ is a subgraph of $G'$ with exactly two $s$-$t$ paths, and routing half a unit of flow along each yields a flow at Nash equilibrium for $(H, r, \ell)$ in which each path has latency 3/2 (cf., Figure 1(b)).

For (ii), we may assume that $H$ contains an $s$-$t$ path. If $H$ has an $s$-$t$ path $P$ containing an $s_2$-$t_1$ path, then routing all of the flow on $P$ yields a Nash flow in which every $s$-$t$ path has latency 2 (cf., Figure 1(a)). Hence, $L(H) = 2$ for all such subgraphs $H$. Otherwise, since $\mathcal{I}$ is a "no" instance of 2DDP, two sole possibilities remain (see Figure 2): either for precisely one $i \in \{1, 2\}$, $H$ has an $s$-$t$ path $P$ containing an $s_i$-$t_i$ path, or all $s$-$t$ paths $P$ in $H$ contain an $s_1$-$t_2$ path of $G$. In either case, routing one unit of flow along such a path $P$ provides a flow at Nash equilibrium showing that $L(H) = 2$. ■

Corollary 3.2 and Theorem 3.3 imply that efficiently detecting Braess's Paradox in networks with linear latency functions is impossible, even in instances suffering from the most severe manifestations of the paradox. To make this statement precise, call an instance $(G, r, \ell)$ with linear latency functions *paradox-free* if $L(G, r, \ell) \leq L(H, r, \ell)$ for all subgraphs $H$ of $G$, and *paradox-ridden* if for some subgraph $H$ of $G$, $L(G, r, \ell) = \frac{4}{3}L(H, r, \ell)$. Paradox-free instances do not suffer from Braess's Paradox and, by Corollary 3.2, paradox-ridden instances are precisely those incurring a worst-possible loss in network performance due to Braess's Paradox. The reduction in the proof of Theorem 3.3 then gives the following corollary.

**Corollary 3.4** *Given an instance $(G, r, \ell)$ that has linear latency functions and is either paradox-free or paradox-ridden, it is NP-hard to decide whether or not $(G, r, \ell)$ is paradox-ridden.*

# 4 General Latency Functions

In this section we study network design with general (continuous, non-decreasing) latency functions. Previous work offers no aid in upper or lower bounding the best approximation ratio that can be obtained for this problem; we will introduce new proof techniques for each of our two (matching) bounds.

In Subsection 4.1, we use the monotone orderings of Subsection 2.5 to prove that the trivial algorithm achieves an approximation ratio of $\lfloor n/2 \rfloor$, where $n$ in the number of vertices in the network. In Subsection 4.2, we show a matching lower bound on the performance guarantee of the trivial algorithm. We accomplish this by giving the first networks that demonstrate that Braess's Paradox can be more severe in larger networks than in Braess's original four-node example [15]. In Subsection 4.3, we use this family of networks to prove an optimal hardness result that matches the performance guarantee of the trivial algorithm.

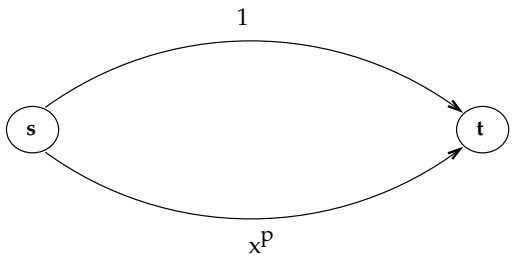



Figure 3: Example 4.1. A foil for the proof approach of Corollary 3.2.

## 4.1 An $\lfloor n/2 \rfloor$-Approximation Algorithm

Our next goal is to prove that the trivial algorithm is an $\lfloor n/2 \rfloor$-approximation algorithm for General Latency Network Design, where $n$ is the number of vertices in the network. Before embarking on the proof, we discuss why this result is not trivial, and does not trivially follow from existing work. In the proof of Corollary 3.2, we saw that an upper bound on the cost of a Nash flow, relative to all other feasible flows, immediately yields an identical upper bound on the approximation ratio of the trivial algorithm. Thus, if we knew that the cost of a Nash flow in a network with $n$ vertices and general latency functions must be at most $\lfloor n/2 \rfloor$ times that of every other feasible flow, then we would be done.[2] Unfortunately, this statement is false, even with $\lfloor n/2 \rfloor$ replaced by an arbitrarily large function of the network size. To see why, we recapitulate an example from [93].

**Example 4.1** We will consider the network shown in Figure 3, with vertices $s$ and $t$ and two edges. The upper and lower edges have latency functions $\ell(x) = 1$ and $\ell(x) = x^p$, respectively, where $p$ is a large integer.

Setting the traffic rate $r$ to 1, the Nash flow routes all traffic on the lower edge and incurs one unit of cost. On the other hand, routing $\epsilon$ units of flow on the upper edge and the rest on the lower edge yields a feasible flow that has near-zero cost when $p$ is sufficiently large and $\epsilon$ is sufficiently small.

Example 4.1 shows that in networks with general latency functions, a Nash flow can be *arbitrarily* more costly than other feasible flows. Moreover, this remains true in a fixed network, even in one with only two nodes and two edges.

However, Example 4.1 is not due cause for abandoning the goal of proving some kind of performance guarantee for the trivial algorithm; it merely indicates that a more delicate approach is required. In Example 4.1, the flow with near-zero cost was far from at equilibrium: a few martyrs were routed on the upper edge for the benefit of the overwhelming majority of the flow. Indeed, for all non-empty subgraphs $H$ of $G$, $L(H) = 1$. Every non-trivial subgraph of $G$ provides an optimal solution to our network design problem, but we cannot prove any finite approximation ratio based on a comparison with arbitrary feasible flows.

We have therefore set the bar unduly high. By comparing the output of the trivial algorithm only to feasible flows at equilibrium in a subgraph of $G$, rather than to *all* feasible

[2]Indeed, this argument will reoccur in Section 5.

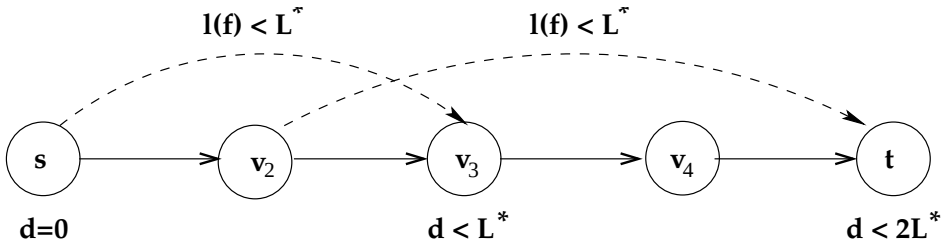



Figure 4: Proof of Theorem 4.2. If $f$ is the flow sending one unit of flow on the four-hop path and $f^*$ is the flow sending half a unit of flow on each of the other two paths, then the dashed edges are light.

flows, we will obtain the main result of this subsection.

**Theorem 4.2** *For every instance $(G, r, \ell)$ with $n$ vertices, the trivial algorithm returns a solution of value at most $\lfloor n/2 \rfloor$ times that of an optimal solution.*

*Proof:* Let $(G, r, \ell)$ be an instance in which $G$ has $n$ vertices. By Proposition 2.4, there is a directed acyclic flow $f$ at Nash equilibrium for $(G, r, \ell)$. Let $H$ be an arbitrary subgraph of $G$ and $f^*$ a Nash flow for $(H, r, \ell)$. Let $d$ and $d^*$ be the distance labels of Definition 2.9 corresponding to the Nash flows $f$ and $f^*$ for the instances $(G, r, \ell)$ and $(H, r, \ell)$, respectively. Let $L^*$ be shorthand for $L(H, r, \ell) = d^*(t)$. We need to show that

$$d(t) \leq \left\lfloor \frac{n}{2} \right\rfloor \cdot L^*. \tag{3}$$

By Proposition 2.11, there is an $f$-monotone ordering of the vertices of $G$. Label these vertices $v_1, v_2, \ldots, v_n$, accordingly. We can assume that $v_1 = s$. Call an edge $e$ of $G$ *light* if $f_e \leq f_e^*$ with $f_e^* > 0$, and *heavy* otherwise. We can finish the proof by establishing the following two claims; see also Figure 4.

(C1) If $v_j$ precedes $t$ in the $f$-monotone ordering, then there is a path of light edges beginning in $\{v_1, v_2, \ldots, v_j\}$ and terminating in $\{t, v_{j+2}, v_{j+3}, \ldots, v_n\}$.

(C2) If there is a path of light edges from $u$ to $v$, then $d(v) \leq d(u) + L^*$.

To see why these two claims imply (3), we first recall that property (P2) from Definition 2.9 states that $d$-values are non-decreasing in the $f$-monotone ordering. Since $d(v_1) = d(s) = 0$, we can apply (C1) and (C2) inductively to the sets $\{v_1, \ldots, v_{2i+1}\}$ to obtain $d(v_{2i+1}) \leq i \cdot L^*$ for $v_{2i+1}$ equal to or preceding $t$. If $t = v_{2i+1}$ for an integer $i$, then (3) follows immediately. If $t = v_{2i}$, then $d(v_{2i-1}) \leq (i-1) \cdot L^*$ and (3) follows from one further application of the two claims to $\{v_1, \ldots, v_{2i-1}\}$.

To prove claim (C1), let $v_j$ precede $t$ in the $f$-monotone ordering. Let $S$ be the $s$-$t$ cut $\{v_1, \ldots, v_j\}$. Adding up the flow conservation constraints for vertices in $S$ implies that the net $f$-flow and $f^*$-flow escaping $S$ is precisely $r$; see, for example, Tarjan [105, Lemma 8.1] for further details. Also, by property (P1) of Definition 2.9, no $f$-flow enters $S$. At least one light edge must therefore escape $S$. If some such edge has its head in $\{t, v_{j+2}, \ldots, v_n\}$, we

are done. If not, all light edges terminate at a vertex $v_{j+1}$ that precedes $t$ in the $f$-monotone ordering. By the above argument, some light edge $e$ escapes $\{v_1, \ldots, v_{j+1}\}$. Since all light edges emanating from $S$ are assumed to end at $v_{j+1}$, the edge $e$ must begin at $v_{j+1}$. Thus, its concatenation with any light edge escaping $S$ provides the desired path of light edges.

For claim (C2), let $P$ be a path of light edges from $u$ to $v$. By the definition of light and the monotonicity of latency functions, $\ell_e(f_e) \leq \ell_e(f_e^*)$ for each edge $e$ in $P$. By the triangle inequality for shortest-path distances (see (2)), we have

$$d(v) - d(u) \leq \sum_{e \in P} \ell_e(f_e) \leq \sum_{e \in P} \ell_e(f_e^*).$$

Since $f_e^* > 0$ on all edges $e$ of $P$, Proposition 2.10 implies that

$$d(v) - d(u) \leq \sum_{e \in P} \ell_e(f_e^*) = d^*(v) - d^*(u).$$

Similarly, by Proposition 2.10,

$$0 \leq d^*(u) \leq d^*(v) \leq L^*.$$

Hence, $d(v) - d(u) \leq d^*(v) - d^*(u) \leq L^*$. This completes the proofs of claim (C2) and the theorem. ■

**Remark 4.3** The special case of Theorem 4.2 for four-node networks was independently obtained by Kameda [56].

## 4.2 The Braess Graphs

In this subsection we will show that the bound of $\lfloor n/2 \rfloor$ in Theorem 4.2 is tight for all $n \geq 2$. This is tantamount to giving a sequence of instances, in which the $n$th instance is a network with $n$ vertices such that removing edges can decrease the latency of traffic in a Nash flow by an $\lfloor n/2 \rfloor$ factor. Prior to this work, it was not known if removing edges in arbitrarily large networks could improve the latency of traffic in a Nash flow by more than a factor of 2.

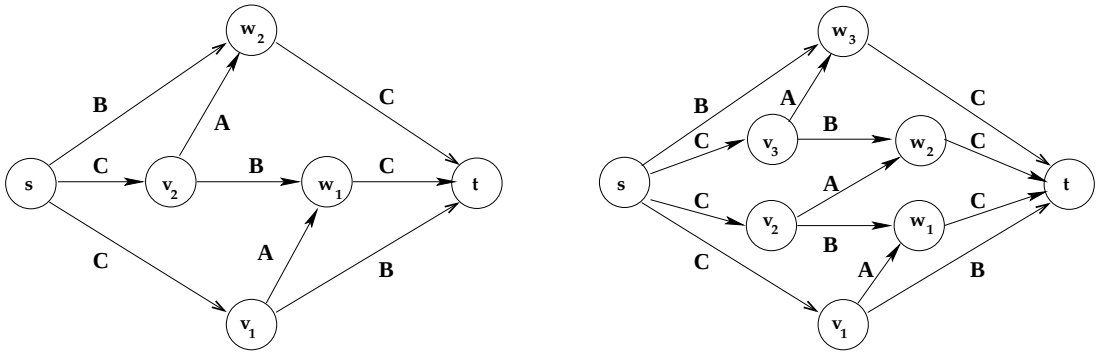
For a positive integer $k$, will we define the *kth Braess graph*. We start with a set $V^k = \{s, v_1, \ldots, v_k, w_1, \ldots, w_k, t\}$ of $2k + 2$ vertices. The edge set $E^k$ is the union of three sets, $\{(s, v_i), (v_i, w_i), (w_i, t) : 1 \leq i \leq k\}$, $\{(v_i, w_{i-1}) : 2 \leq i \leq k\}$, and $\{(v_1, t)\} \cup \{(s, w_k)\}$ (see Figure 5). We note that $B^1$ is the graph in which Braess's Paradox was first discovered (Figure 1(a)).

We next define latency functions $\ell^k$ for the edges of $B^k$; these functions will prove useful in Proposition 4.4 below. See also Figure 5.

(A) Edges of the form $e = (v_i, w_i)$ are given the latency function $\ell_e^k(x) = 0$.

(B) For an edge $e$ of the form $(v_i, w_{i-1})$, $(s, w_k)$, or $(v_1, t)$, put $\ell_e^k(x) = 1$.

(C) For each $i \in \{1, 2, \ldots, k\}$, the edges $(w_i, t)$ and $(s, v_{k-i+1})$ each receive a continuous, non-decreasing latency function $\ell_e^k(x)$ with $\ell_e^k(k/(k+1)) = 0$ and $\ell_e^k(1) = i$.




(a) $B^2$ (b) $B^3$

Figure 5: The second and third Braess graphs. Edges are labeled with their type.

We will call edges of the form $(v_i, w_i)$ *type A edges*, and so forth. The latency functions of type C edges can be defined arbitrarily, except for at the two specified points.

We can now show that the Braess graphs are the promised family of instances in which Braess's Paradox can be arbitrarily severe, or equivalently on which the trivial algorithm performs badly.

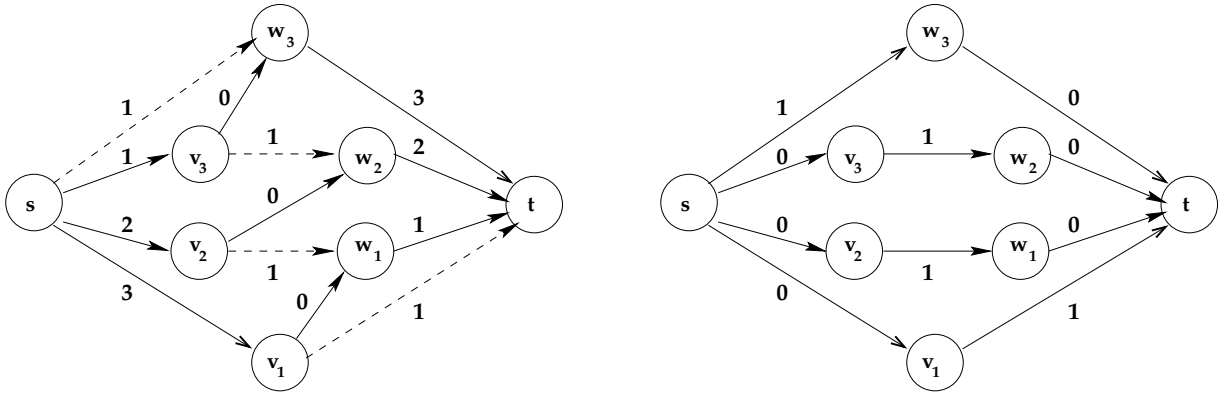
**Proposition 4.4** *For every integer $n \geq 2$, there is an instance $(G, r, \ell)$ in which $G$ has $n$ vertices and a subgraph $H$ with*

$$L(G, r, \ell) = \left\lfloor \frac{n}{2} \right\rfloor \cdot L(H, r, \ell).$$

*Proof:* We can assume that $n$ is even, since the odd case can be reduced to the even case by adding an isolated vertex. We can also assume that $n$ is at least 4. Write $n = 2k + 2$ for a positive integer $k$ and consider the instance $(B^k, k, \ell^k)$. For $i = 1, \ldots, k$, let $P_i$ denote the path $s \to v_i \to w_i \to t$. For $i = 2, \ldots, k$, let $Q_i$ denote the path $s \to v_i \to w_{i-1} \to t$. Define $Q_1$ to be the path $s \to v_1 \to t$ and $Q_{k+1}$ the path $s \to w_k \to t$. On one hand, routing one unit of flow on each of $P_1, \ldots, P_k$ yields a flow at Nash equilibrium for $(B^k, k, \ell^k)$ demonstrating that $L(B^k, k, \ell^k) = k + 1$ (Figure 6(a)). On the other hand, if $H$ is the subgraph obtained from $B^k$ by deleting the $k$ type $A$ edges, then routing $k/(k+1)$ units of flow on each of $Q_1, \ldots, Q_{k+1}$ yields a flow at Nash equilibrium for $(H, k, \ell^k)$ showing that $L(H, k, \ell^k) = 1$ (Figure 6(b)). Thus, $L(G)/L(H) = k + 1 = n/2$, completing the proof. ■

**Corollary 4.5** *For every integer $n \geq 2$, there is an instance $(G, r, \ell)$ in which $G$ has $n$ vertices and for which the trivial algorithm produces a solution with value $\lfloor n/2 \rfloor$ times that of an optimal solution.*

**Remark 4.6** Proposition 4.4 was previously known only for networks with at most four nodes [56, 95].




(a) Nash flow for $(B^3, 3, \ell^3)$

(b) Nash flow in the optimal subgraph

Figure 6: Proof of Proposition 4.4, when $k = 3$. Solid edges carry flow in the flow at Nash equilibrium, dashed edges do not. Edge latencies are with respect to flows at Nash equilibrium.

**Remark 4.7** In the proof of Proposition 4.4, we removed $k$ edges from $(B^k, k, \ell^k)$ to decrease the latency encountered by traffic in a Nash flow by a factor of $k + 1$. Braess [15] removed but a single edge in his seminal example. Recently, Lin, Roughgarden, and Tardos [72] demonstrated the necessity of removing multiple edges, in the following sense: for all $k \geq 1$, the only way to decrease the latency of traffic in a Nash flow by a factor strictly larger than $k$ is to remove at least $k$ edges from the network.

## 4.3 Proof of Hardness

We now show that the trivial algorithm is the best-possible approximation algorithm for General Latency Network Design, in the sense that no $(\lfloor n/2 \rfloor - \epsilon)$-approximation algorithm exists, assuming $P \neq NP$. We begin with an informal description of the reduction. Recall that in an instance of the NP-complete problem Partition, we are given $p$ positive integers $\{a_1, a_2, \ldots, a_p\}$ and seek a subset $S \subseteq \{1, 2, \ldots, p\}$ such that $\sum_{j \in S} a_j = \frac{1}{2} \sum_{j=1}^{p} a_j$ [49, SP12]. The idea of the reduction is to start with a Braess graph, and to replace the type A edges of the form $(v_i, w_i)$ with a collection of parallel edges representing an instance $\mathcal{I} = \{a_1, \ldots, a_p\}$ of Partition. We will endow these edges with latency functions that simulate "capacities", with an edge representing an integer $a_j$ of $\mathcal{I}$ receiving capacity $a_j$. We will then formalize the following three ideas. First, if too many edges are removed from the network, there will be insufficient remaining capacity to send flow cheaply. On the other hand, if too few edges are removed, the excess of capacity results in a Nash equilibrium similar to that of Figure 6(a). Finally, these two cases can be avoided if and only if $\mathcal{I}$ is a "yes" instance of Partition, in which case removing the appropriate collection of edges results in a network that admits a Nash equilibrium similar to that of Figure 6(b).

**Theorem 4.8** *Assuming $P \neq NP$, for every $\epsilon > 0$ there is no $(\lfloor n/2 \rfloor - \epsilon)$-approximation algorithm for* General Latency Network Design.

*Proof:* We prove that for every fixed $n \geq 2$, there is no $(\lfloor n/2 \rfloor - \epsilon)$-approximation algorithm for General Latency Network Design restricted to (multi)graphs with $n$ vertices. We will prove this via a "gap" reduction from the NP-complete problem Partition.

Fix $n \geq 2$. As in the proof of Proposition 4.4, we can assume that $n$ is even and at least four. Write $n = 2k + 2$ for a positive integer $k$. Consider an instance $\mathcal{I} = \{a_j\}_{j=1}^p$ of Partition, with each $a_j$ a positive integer. By scaling, we can assume that every $a_j$ is a multiple of 3. Put $A = \sum_{j=1}^p a_j$. We will be interested in the traffic rate

$$r = k\frac{A}{2} + k + 1. \tag{4}$$

Construct a graph $G$ from the $k$th Braess graph $B^k$ by replacing each edge of the form $(v_i, w_i)$ by $p$ parallel edges, and denote these by $e_i^1, e_i^2, \ldots, e_i^p$.

We now specify the edge latency functions $\ell$, which are more complicated than in the previous subsection. We will require a sufficiently small constant $\delta$ and a sufficiently large constant $M$. The following proof will be valid if we take $\delta = 1/A(p+k)$ and $M = n/2$. The constant $M$ should be interpreted as a substitute for $+\infty$, and is used to penalize a flow for violating an edge capacity constraint. We require the constant $\delta$ to transform step functions—the type of function that would be most convenient for our argument—into continuous functions, the type of function allowed in our model. The parameter $\delta$ provides a small window in which to "smooth out" the discontinuities of a step function. We now define the latency functions.

(A) An edge of the form $e_i^j$ is given a latency function $\ell$ with $\ell(x) = 0$ for $x \leq a_j - \delta$, $\ell(a_j) = 1$, and $\ell(x) = M$ for $x \geq a_j + \delta$.

(B) Edges of the form $(v_i, w_{i-1})$, $(s, w_k)$, or $(v_1, t)$ receive a latency function $\ell$ satisfying $\ell(x) = 1$ for $x \leq 1$ and $\ell(x) = M$ for $x \geq 1 + \delta$.

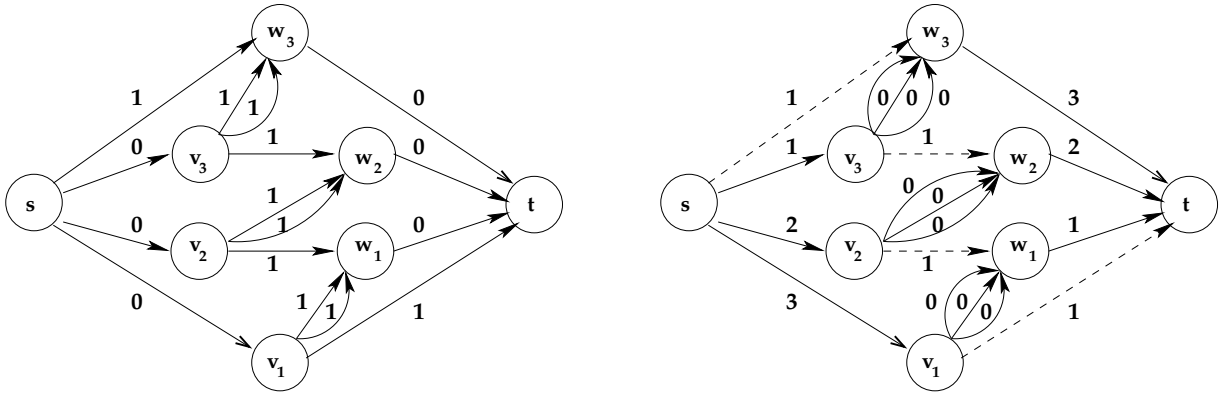
(C) For each $i \in \{1, 2, \ldots, k\}$, the two edges $(w_i, t)$ and $(s, v_{k-i+1})$ are given a latency function $\ell$ satisfying $\ell(x) = 0$ for $x \leq A/2 + 1$, $\ell(x) = i$ when $x = A/2 + (k+1)/k$, and $\ell(x) = M$ for $x \geq A/2 + (k+1)/k + \delta$.

As usual, these latency functions can be defined arbitrarily outside of the regions that we have prescribed, subject to the usual continuity and monotonicity constraints. The instance $(G, r, \ell)$ can be constructed in time polynomial in the size of the Partition instance $\mathcal{I}$. We will say that an edge of the form $e_i^j$ has *capacity* $a_j$, while edges of type B and C have capacity 1 and $A/2 + (k+1)/k$, respectively. We will call an edge *oversaturated* by a flow if the amount of flow on it exceeds its capacity be at least an additive factor of $\delta$. An oversaturated edge therefore has latency $M$.

Analogous to the proof of Theorem 3.3, it suffices to prove the following two statements.

(i) If $\mathcal{I}$ is a "yes" instance of Partition, then $G$ admits a subgraph $H$ with $L(H, r, \ell) = 1$.

(ii) If $\mathcal{I}$ is a "no" instance, then $L(H, r, \ell) \geq n/2$ for every subgraph $H$ of $G$.




(a) A good Nash flow corresponding to a "yes" instance of PARTITION, with $m = 2$.

(b) A bad Nash flow in a network with excess capacity.

Figure 7: Proof of Theorem 4.8. Solid edges carry flow in the flow at Nash equilibrium, dashed edges do not. Edge latencies are with respect to flows at Nash equilibrium.

To prove (i), suppose that $\mathcal{I}$ admits a partition, and reindex the $a_j$'s so that $\sum_{j=1}^{m} a_j = A/2$ for some $m \in \{1, 2, \ldots, p-1\}$. Obtain $H$ from $G$ by deleting all edges of the form $e_i^j$ for $j > m$. For each $i = 1, \ldots, k$, the remaining edges of the form $e_i^j$ have total capacity $A/2$. Define the paths $Q_1, \ldots, Q_{k+1}$ as in the proof of Proposition 4.4: for $i = 2, \ldots, k$, $Q_i$ denotes the path $s \to v_i \to w_{i-1} \to t$, $Q_1$ is the path $s \to v_1 \to t$, and $Q_{k+1}$ is the path $s \to w_k \to t$. Define a feasible flow $f$ as follows: for each $i = 1, \ldots, k$ and $j = 1, \ldots, m$, route $a_j$ units of flow on the unique path containing edge $e_i^j$, and route 1 unit of flow on the path $Q_i$ for $i = 1, 2, \ldots, k+1$. The flow $f$ is at Nash equilibrium for $(H, r, \ell)$ and proves that $L(H, r, \ell) = 1$ (see Figure 7(a)).

Statement (ii) is more difficult. We first recall that if a flow oversaturates an edge, then the edge has latency $M$. A simple but crucial observation is that if a Nash flow in the instance $(H, r, \ell)$ oversaturates an edge, then $L(H, r, \ell) \geq M \geq n/2$. We will prove (ii) in two steps. First, we will identify two sufficient conditions on the subgraph $H$ that ensure that a Nash flow in $(H, r, \ell)$ must oversaturate some edge and hence $L(H, r, \ell) \geq n/2$. Then, we will give a separate argument that $L(H, r, \ell) \geq n/2$ for subgraphs $H$ that do not meet these sufficient conditions.

We first claim that if a subgraph $H$ omits some type C edge, then a Nash flow in $(H, r, \ell)$ must oversaturate an edge. To see why, we observe that if $H$ omits a type C edge incident to $s$, then the remaining capacity on edges incident to $s$ is at most that of one type B edge and $k-1$ type C edges, which is

$$1 + (k-1)\left(\frac{1}{2}A + \frac{k+1}{k}\right) < (k-1)\frac{A}{2} + k + 1. \tag{5}$$

Since the right-hand side of (5) is at most the traffic rate (4), every feasible flow in $(H, r, \ell)$ must oversaturate some edge incident to $s$, provided $\delta$ is sufficiently small. The same argu-

ment shows that if $H$ omits a type C edge incident to $t$, then every feasible flow for $(H, r, \ell)$ oversaturates some edge.

Next, we claim that if for some $i \in \{1, \ldots, k\}$, the total capacity $A_i$ of edges of the form $e_i^j$ in $H$ is less than $A/2$, then every feasible flow in $(H, r, \ell)$ oversaturates an edge. To see this, suppose $A_i < A/2$. Since all $a_j$'s are multiples of 3, we must then have $A_i \leq A/2 - 3$. Besides edges of the form $e_i^j$, the only edge leaving the vertex $v_i$ is the unit-capacity type B edge $(v_i, w_{i-1})$ (or $(v_1, t)$, if $i = 1$). The total capacity out of the vertex $v_i$ is therefore at most $A/2 - 2$. This in turn gives the edge $(s, v_i)$ an "effective capacity" of $A/2 - 2$, in the sense that routing at least $A/2 - 2 + p\delta$ units of flow on $(s, v_i)$ must oversaturate some edge out of $v_i$. The effective capacity of edges incident to $s$ is thus only $A/2 - 2$ more than in the calculation in (5), which is at most

$$k\frac{A}{2} + k - 1 = r - 2.$$

As in the previous paragraph, for $\delta$ sufficiently small, every feasible flow in $(H, r, \ell)$ must oversaturate some edge.

Finally, suppose that the subgraph $H$ contains all type C edges and that the total capacity $A_i$ of edges of the form $e_i^j$ in $H$ is at least $A/2$ for every $i$. Since $\mathcal{I}$ is a "no" instance of PARTITION and each $a_j$ is a multiple of 3, $A_i \geq A/2 + 3$ for every $i$. Then, define a flow $f$ in $H$ as follows: for each $i = 1, \ldots, k$ and $j = 1, \ldots, p$ such that $e_i^j$ is present in $H$, route

$$\frac{a_j}{A_i}\left(\frac{A}{2} + \frac{k+1}{k}\right)$$

units of flow along the unique $s$-$t$ path containing $e_i^j$. Since $A_i \geq A/2 + 3$ for every $i$ and $k \geq 1$, the latency of every edge of the form $e_i^j$ in $H$ is zero with respect to $f$. The flow $f$ is at Nash equilibrium and proves that $L(H) = n/2$, see Figure 7(b). ■

As in Corollary 3.4, the matching upper and lower bounds of Theorems 4.2 and 4.8 have strong negative consequences for the problem of detecting Braess's Paradox. Defining an instance $(G, r, \ell)$ with general latency functions and $n$ vertices to be *paradox-free* if $L(G, r, \ell) \leq L(H, r, \ell)$ for all subgraphs $H$ of $G$ and *paradox-ridden* if for some subgraph $H$ of $G$, $L(G, r, \ell) = \lfloor n/2 \rfloor \cdot L(H, r, \ell)$, we obtain the following corollary.

**Corollary 4.9** *Given an instance $(G, r, \ell)$ with general latency functions that is paradox-free or paradox-ridden, it is NP-hard to decide whether or not $(G, r, \ell)$ is paradox-ridden.*

**Remark 4.10** We will obtain a hardness result that is similar but incomparable to Theorem 4.8 as a consequence of our work in the next section. See Corollary 5.7 and Remark 5.8 for details.

# 5 Extensions

In this section, we aim to show that the strong hardness results of Sections 3 and 4 extend beyond the particular classes of linear and general latency functions, and seem intrinsic to

the problem of designing networks for selfish users. In Subsection 5.1 we consider a natural extension of the linear latency setting, where all latency functions are polynomials with bounded degree and non-negative coefficients. In Subsection 5.2, we generalize our results still further.

## 5.1 Polynomials of Bounded Degree

As in Section 3, we begin by observing that existing work bounding the worst-case inefficiency of flows at Nash equilibrium yields an upper bound on the performance guarantee of the trivial algorithm. The following result is due to Roughgarden [90].

**Proposition 5.1 ([90])** *Let $p$ be a positive integer and $(G, r, \ell)$ an instance in which every latency function has the form $\ell(x) = a_p x^p + a_{p-1}x^{p-1} + \cdots + a_0$, with $a_i \geq 0$ for each $i$. If $f^*$ is a feasible flow and $f$ is a flow at Nash equilibrium for $(G, r, \ell)$, then*

$$C(f) \leq (1 - p \cdot (p+1)^{-(p+1)/p})^{-1} \cdot C(f^*).$$

Example 4.1 and a back-of-the-envelope calculation show that the upper bound in Proposition 5.1 cannot be improved.

We will say that an instance with latency functions as in Proposition 5.1 has *polynomial latency functions of degree $p$*, with the understanding that all coefficients are non-negative. We will call the corresponding network design problem Polynomial($p$) Latency Network Design. For clarity, we will work with the following asymptotic form of Proposition 5.1.

**Corollary 5.2** *There is a constant $c_1 > 0$ such that for every $p \geq 2$ and every instance $(G, r, \ell)$ with polynomial latency functions of degree $p$ that admits a Nash flow $f$ and a feasible flow $f^*$,*

$$C(f) \leq c_1 \frac{p}{\ln p} \cdot C(f^*).$$

As in Corollary 3.2, we immediately obtain an upper bound on the performance guarantee of the trivial algorithm.

**Corollary 5.3** *There is a constant $c_1 > 0$ such that, for every $p \geq 2$, the trivial algorithm is a $c_1 \frac{p}{\ln p}$-approximation algorithm for* Polynomial($p$) Latency Network Design.

We next work toward a proof of a matching hardness result. As in Section 4, we first give a family of networks on which the trivial algorithm performs poorly, and then describe how to obtain a general inapproximability result.

**Proposition 5.4** *There is a constant $c_2 > 0$ such that, for every integer $p \geq 2$, there is an instance $(G, r, \ell)$ with polynomial latency functions of degree $p$ for which the trivial algorithm produces a solution with value at least $c_2 \frac{p}{\ln p}$ times that of an optimal solution.*

*Proof:* We will again make use of the Braess graphs of Subsection 4.2. In the proof of Proposition 4.4, we exploited the fact that general latency functions can be arbitrarily steep

to construct a bad example for the trivial algorithm. Here, we will adapt the previous argument as best we can, given that only low-degree polynomials are at our disposal.

For each integer $p \geq 2$, we will define a parameter $k$ and a set of latency functions $\ell^k$ for the edges of $B^k$. We will choose the parameter $k$ later. The functions $\ell^k$ are identical to those in Subsection 4.2, except that for every $i \in \{1, 2, \ldots, k\}$, the type C edges $(s, v_{k-i+1})$ and $(w_i, t)$ are given the latency function $\ell^k(x) = ix^p$. Next, consider the instance $(B^k, k, \ell^k)$ and define paths $P_1, \ldots, P_k$ and $Q_1, \ldots, Q_{k+1}$ as in the proof of Proposition 4.4. On one hand, routing one unit of flow on each of $P_1, \ldots, P_k$ yields a flow at Nash equilibrium for $(B^k, k, \ell^k)$ showing that $L(B^k, k, \ell^k) = k + 1$, as in Figure 6(a). On the other hand, if $H$ is the subgraph obtained from $B^k$ by deleting all edges of the form $(v_i, w_i)$, routing $k/(k+1)$ units of flow on each of $Q_1, \ldots, Q_{k+1}$ yields a flow at Nash equilibrium for $(H, k, \ell^k)$ showing that

$$L(H, k, \ell^k) = 1 + k\left(\frac{k}{k+1}\right)^p \leq 1 + ke^{-p/(k+1)},$$

as in Figure 6(b). Putting $k = \lfloor \frac{p}{2 \ln p} \rfloor - 1$ we obtain $L(H, k, \ell^k) \leq 2$ and $L(B^k, k, \ell^k) = \lfloor \frac{p}{2 \ln p} \rfloor$. Since $p \geq 2$ was arbitrary, the proof is complete. ■

**Remark 5.5** In the proof of Proposition 5.4, we have avoided optimizing constants for the sake of readability. We will continue to make this tradeoff in the rest of the paper.

Finally, we extend our lower bound on the performance guarantee of the trivial algorithm to an inapproximability result. This task is more difficult than in Section 4, as a crucial part of the proof of Theorem 4.8 leveraged the fact that general latency functions can model edge capacities. This is not entirely possible with low-degree polynomials, and we are forced instead to adapt the arguments of Section 3 to larger Braess graphs. In particular, our reduction is from the 2 Directed Disjoint Paths problem rather than from Partition. In essence, restricting the allowable latency functions forces us to encode the intractability of an NP-hard problem into the network topology of a network design instance rather than into the edge latency functions.

**Theorem 5.6** *Assuming $P \neq NP$, there is a constant $c_3 > 0$ such that for every $p \geq 2$, there is no $c_3 \frac{p}{\ln p}$-approximation algorithm for* Polynomial($p$) Latency Network Design.

*Proof:* Define $k$ by

$$k = \left\lfloor \frac{p}{16 \ln p} \right\rfloor - 1. \tag{6}$$

We can assume that $p$ is sufficiently large, and hence $k \geq 1$. We will show that a $(k/6)$-approximation algorithm for Polynomial($p$) Latency Network Design enables us to differentiate between "yes" and "no" instances of the 2 Directed Disjoint Paths (2DDP) problem in polynomial time.

Let $\mathcal{I} = \{G, s_1, s_2, t_1, t_2\}$ be an instance of 2DDP. We will construct an instance of Polynomial($p$) Latency Network Design $(G', k, \ell)$ as follows; see also Figure 8. The graph $G'$ comprises $k$ copies of $G$, that we will call $G_1, \ldots, G_k$. We will denote the copy of $s_i$ and $t_i$ in $G_j$ by $s_i^j$ and $t_i^j$, respectively. Next, we add auxiliary vertices $s$, $t$, $v_1, \ldots, v_{k-1}$, and $w_1, \ldots, w_{k-1}$. The edge set of $G'$ is as follows:

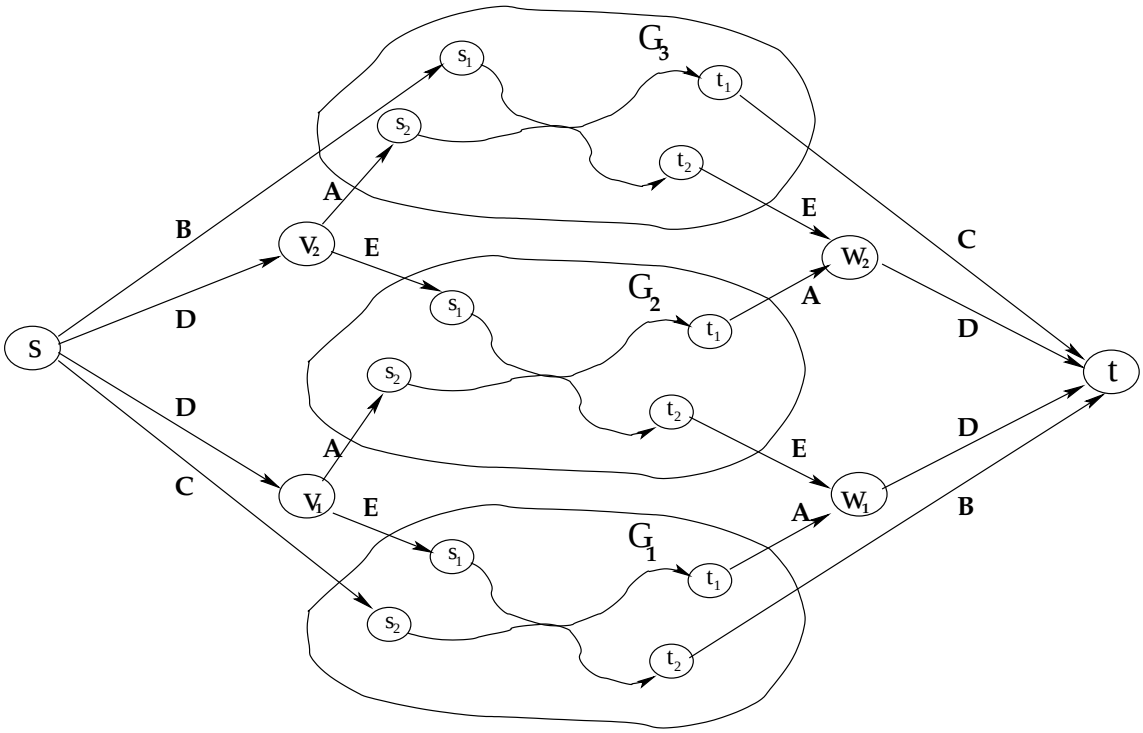



Figure 8: Proof of Theorem 5.6. Construction of $(G', k, \ell)$ when $k = 3$. Edges are labeled with their type.

- each $G_i$ inherits the edge set of $G$;
- for $i = 1, \ldots, k-1$, we include edges from $s$ to $v_i$, from $v_i$ to $s_1^i$ and $s_2^{i+1}$, from $t_1^i$ and $t_2^{i+1}$ to $w_i$, and from $w_i$ to $t$;
- we include edges $(s, s_2^1)$, $(s, s_1^k)$, $(t_2^1, t)$, and $(t_1^k, t)$.

We define latency functions on the edges of $G'$ as follows:

(A) for edges of the form $(v_i, s_2^{i+1})$ or $(t_1^i, w_i)$, put $\ell(x) = 1$;

(B) for edges $(s, s_1^k)$ and $(t_2^1, t)$, put $\ell(x) = 2 + (1 + \frac{1}{k})^p x^p$;

(C) for $(s, s_2^1)$ and $(t_1^k, t)$, put $\ell(x) = 1 + k(\frac{4(k+1)}{4k+1})^p x^p$;

(D) for $i = 1, \ldots, k-1$ and edges $(s, v_{k-i})$ and $(w_i, t)$, put $\ell(x) = i(\frac{4(k+1)}{4k+1})^p x^p$;

(E) for edges of the form $(v_i, s_1^i)$ or $(t_2^{i+1}, w_i)$, put $\ell(x) = 2 + (2 + \frac{2}{k})^p x^p$;

(F) for edges in $G_1, \ldots, G_k$, put $\ell(x) = 0$.

The instance $(G', k, \ell)$ can be constructed from $\mathcal{I}$ in polynomial time.

Next, we claim that if $\mathcal{I}$ is a "yes" instance of 2DDP, then there is a subgraph $H$ of $G'$ with $L(H, k, \ell) \leq 5$. To see why, let $P_1^*$ and $P_2^*$ denote vertex-disjoint $s_1$-$t_1$ and $s_2$-$t_2$ paths in $G$, respectively. Deleting all edges in $G'$ that lie in some copy of $G$ but not on (the corresponding copy of) either $P_1^*$ or $P_2^*$, we obtain a subgraph $H$ of $G'$ that is the union of $2k$ distinct $s$-$t$ paths. Let $f$ be the flow that routes $k/(k+1)$ units of flow on both the path containing $s_2^1$ and $t_2^1$ and on the path containing $s_1^k$ and $t_1^k$, and $k/2(k+1)$ units of flow on each of the other $2k-2$ paths. Since each type D edge lies in two paths of the latter type, all edges incident to $s$ or $t$ carry $k/(k+1)$ units of flow. Somewhat analogous to Figure 6(b), $f$ is at Nash equilibrium for $(H, k, \ell)$ and proves that

$$L(H, k, \ell) = 4 + k\left(\frac{4(k+1)}{4k+1}\frac{k}{k+1}\right)^p = 4 + k\left(1 - \frac{1}{4k+1}\right)^p \leq 4 + ke^{-p/(4k+1)} \leq 5,$$

where the last inequality follows from (6).

Finally, we show that if $\mathcal{I}$ is a "no" instance of 2DDP, then $L(H, k, \ell) \geq k$ for all subgraphs $H$ of $G'$. We will prove this in two steps. First, we will show that unless $H$ contains most of the edges in $G'$, capacity considerations in the spirit of the proof of Theorem 4.8 imply that $L(H)$ is large. Second, we will show that if $H$ contains most of the edges in $G'$, then the flow at Nash equilibrium in $H$ is similar to the bad Nash flow of Proposition 5.4, again showing that $L(H)$ is large.

We first claim that if a subgraph $H$ omits a type A or C edge of $G'$, then $L(H) \geq k$. We will prove this claim for an edge of the form $(v_i, s_2^{i+1})$. The argument for an edge of the form $(t_1^i, w_i)$ is symmetric, and the argument for type C edges is both similar and easier.

To prove this claim, we first observe that many edges of $H$ are essentially capacitated, in the following sense. We assert that if a flow $f$ causes one of the following three events, then the corresponding edge $e$ has latency at least $p$:

(1) $f_e \geq \frac{8k+1}{8(k+1)}$ for a type B edge $e$;

(2) $f_e \geq \frac{2k+1}{2(k+1)}$ for an edge $e$ of type C or D;

(3) $f_e \geq \frac{4k+1}{8(k+1)}$ for a type E edge $e$.

For example, using (6) we can derive

$$\left(\frac{4(k+1)}{4k+1}\frac{2k+1}{2(k+1)}\right)^p = \left(1 + \frac{1}{4k+1}\right)^p > \left[e^{1/(8k+2)}\right]^p = e^{p/(8k+2)} \geq p,$$

which proves the assertion for events of type (2). The calculations for the other two types of events are similar, so we omit them.

Now assume that the edge $(v_i, s_2^{i+1})$ is absent from the subgraph $H$ or $G'$. Then, every flow feasible for $(H, k, \ell)$ must either route at most $(4k+1)/8(k+1)$ units of flow on the edge $(s, v_i)$, or else must cause event (3) to occur with the other edge leaving $v_i$, the type E edge $(v_i, s_1^i)$. We claim if at most $(4k+1)/8(k+1)$ units of flow are routed on the edge $(s, v_i)$, then event (1) or event (2) must occur with some edge incident to $s$. For if not, edges incident to $s$ carry at most

$$(k-1)\frac{2k+1}{2(k+1)} + \frac{4k+1}{8(k+1)} + \frac{8k+1}{8(k+1)} = \frac{8k^2+8k-2}{8(k+1)} < k \tag{7}$$

units of flow. The first term on the left-hand side of (7) is for edges of types C or D other than the edge $(s, v_i)$, the second term is for the edge $(s, v_i)$, and the third term is for the type B edge $(s, s_1^k)$. Inequality (7) provides a contradiction, since a flow feasible for $(H, k, \ell)$ must route $k$ units of flow out of the source $s$. Since a Nash flow for $(H, k, \ell)$ must cause some event of the form (1), (2), or (3) to occur, $L(H) \geq p \geq k$.

We now consider the remaining case and make use of our hypothesis that $\mathcal{I}$ is a "no" instance of 2DDP. Let $H$ be a subgraph of $G$, and $f$ a Nash flow for $(H, k, \ell)$. By our earlier arguments, $L(H) \geq k$ unless $f$ routes flow on all type A and C edges. If $f$ routes flow on all type A and C edges, the construction of $G'$ ensures that for all $i \in \{1, 2, \ldots, k\}$, the vertices $s_2^i$ and $t_1^i$ lie on $s$-$t$ paths in $H$. Since $\mathcal{I}$ is a "no instance", for each $i \in \{1, 2, \ldots, k\}$ there must then be an $s$-$t$ path $P_i$ containing both $s_2^i$ and $t_1^i$ (see Figure 2). Routing $(4k+1)/4(k+1)$ units of flow on each path $P_i$ defines a flow at Nash equilibrium for $(H, r, \ell)$, where

$$r = \frac{k(4k+1)}{4(k+1)} < k.$$

As in Proposition 5.4, this flow shows that $L(H, r, \ell) = k + 3$. By Proposition 2.8, we then have $L(H, k, \ell) \geq k + 3$.

We have exhausted all possible cases, and the proof is complete. ■

We now fulfill a promise made in Remark 4.10.

**Corollary 5.7** *Assuming $P \neq NP$, for every $\epsilon > 0$ there is no $O(n^{1-\epsilon})$-approximation algorithm for* General Latency Network Design.

*Proof:* Fix $\epsilon > 0$, and choose a positive integer $m \geq 1/\epsilon$. Let $\mathcal{I} = \{G, s_1, s_2, t_1, t_2\}$ be an instance of 2DDP. Let $q$ be the number of vertices of $G$. We can assume that $G$ has no parallel arcs [44], and hence the size of $\mathcal{I}$ is polynomial in $q$.

Next, we mimic the reduction in the proof of Theorem 5.6, setting the parameter $k$ in the construction to $q^m$. We also set the parameter $p = \Theta(q^m \log q)$ accordingly. The reduction creates an instance $(G', k, \ell)$, where $G'$ has $n = q^{m+1} + 2q^m$ vertices and the functions $\ell$ are polynomials with degree at most $p$ and coefficients of moderate size. Since $m$ is fixed, the instance $(G', k, \ell)$ can be constructed in time polynomial in $q$. The construction also ensures that if $\mathcal{I}$ is a "yes" instance of 2DDP, then there is a subgraph $H$ of $G'$ with $L(H, k, \ell) \leq 5$, while if $\mathcal{I}$ is a "no" instance then $L(H, k, \ell) \geq cq^m$ for some constant $c > 0$ and all subgraphs $H$ of $G'$. Since $G'$ has $O(q^{m+1})$ vertices, this precludes an $o(n^{1-1/(m+1)})$-approximation algorithm for General Latency Network Design, assuming $P \neq NP$. Since $1/(m+1) < \epsilon$ and $\epsilon > 0$ was arbitrary, the corollary follows. ■

**Remark 5.8** Corollary 5.7 is inferior to Theorem 4.8 in an obvious sense: it proves a weaker lower bound on the performance guarantee for General Latency Network Design that is achievable by polynomial-time algorithms. On the other hand, Corollary 5.7 is superior to Theorem 4.8 in two respects. First, it applies to an easier problem, in which graphs are restricted to be simple (without parallel edges), and latency functions can only be polynomials with non-negative coefficients. Second, the reduction in the proof of Corollary 5.7 is from the strongly NP-complete problem 2DDP, while the reduction in the proof of Theorem 4.8 is from the weakly NP-complete problem Partition. See Garey and Johnson [49] for details on this distinction.

## 5.2 Further Extensions

In this subsection we accumulate further evidence that the intractability of designing networks for selfish users is not sensitive to the allowable latency functions. Before introducing the final set of latency functions that we will consider, we note that identifying a large set of latency functions that behave better than general ones is a non-trivial task. For example, one natural idea is to require that network latency functions possess one or more bounded derivatives on the domain of interest, $[0, r]$. However, every instance $(G, r, \ell)$ with $C^k$ latency functions—latency functions that are $k$ times continuously differentiable—can be "scaled down" to an instance $(G, r, \frac{1}{M}\ell)$ in which the first $k$ derivatives of all edge latency functions are as small as desired, by taking $M$ sufficiently large. Moreover, the network design problem on the scaled instance is equivalent to that on the original instance. Theorem 4.8 has therefore not been avoided, and for every $\epsilon > 0$, no $(\lfloor n/2 \rfloor - \epsilon)$-approximation algorithm exists for such instances, assuming $P \neq NP$.

We thus require some "scale-invariant" parameter ensuring that a latency function will behave better than an arbitrary one. Toward this end, for an instance $(G, r, \ell)$ and an edge $e$ of $G$, define the quantity $\Gamma_e(x)$ by

$$\Gamma_e(x) = \begin{cases} \frac{x \cdot \ell_e(x)}{\int_0^x \ell_e(y)\, dy} & \text{if } \ell_e(x) > 0 \text{ and } x > 0 \\ 1 & \text{otherwise.} \end{cases}$$

Since every latency function $\ell_e$ is non-decreasing, $\Gamma_e(x) \geq 1$ for all $e \in E$ and $x \geq 0$. Define $\Gamma(G, r, \ell)$ by

$$\Gamma(G, r, \ell) = \max_{e \in E} \sup_{x \in [0,r]} \Gamma_e(x).$$

For example, the $\Gamma$-value of an instance with polynomial latency functions of degree $k$ is at most $k + 1$.

These bizarre definitions are justified by the next result, due to Roughgarden and Tardos [93], which bounds the inefficiency of a flow at Nash equilibrium in an instance $(G, r, \ell)$ by $\Gamma(G, r, \ell)$.

**Proposition 5.9 ([93])** *Suppose $f$ and $f^*$ are Nash and feasible flows, respectively, for an instance $(G, r, \ell)$. Then,*

$$C(f) \leq \Gamma(G, r, \ell) \cdot C(f^*).$$

Proposition 5.9 follows from the discussion following Proposition 2.3, which asserts that Nash flows are the optima of a certain convex program [13], and the fact that the objective function of this convex program differs from the cost by at most a $\Gamma(G, r, \ell)$ factor. We refer the reader to [93] for details.

While Proposition 5.9 is not as strong as Propositions 3.1 and 5.1 in the special case of networks with polynomial latency functions, it yields non-trivial upper bounds on the worst-case inefficiency of Nash flows for a broad spectrum of latency functions. As usual, we obtain an upper bound on the performance guarantee of the trivial algorithm as a corollary.

**Corollary 5.10** *The trivial algorithm is a $\gamma$-approximation algorithm for network design instances $(G, r, \ell)$ satisfying $\Gamma(G, r, \ell) \leq \gamma$.*

Up to a constant factor, the upper bound in Corollary 5.10 is the best possible.

**Proposition 5.11** *For every $\gamma \geq 1$, there is an instance $(G, r, \ell)$ with $\Gamma(G, r, \ell) \leq \gamma$ for which the trivial algorithm produces a solution with value $\gamma/2$ times that of an optimal solution.*

The proof of Proposition 5.11 is the same as that for Proposition 5.4, except for two modifications. First, the parameter $k$, which controls the size of the underlying Braess graph, is set to $\lfloor \gamma \rfloor$. Second, the latency functions for the type C edges $(s, v_{k-i+1})$ and $(w_i, t)$ are changed from $ix^p$ to the function that is equal to $i/\gamma$ on $[0, \gamma/(\gamma+1)]$, and is linear on $[\gamma/(\gamma+1), 1]$, subject to $g(\gamma/(\gamma+1)) = i/\gamma$ and $g(1) = i$.

Moreover, we can extend the lower bound on the approximation ratio of the trivial algorithm to an inapproximability result.

**Theorem 5.12** *There is a constant $c > 0$ such that for all $\gamma \geq 1$, there is no $(c \cdot \gamma)$-approximation algorithm for network design for instances $(G, r, \ell)$ with $\Gamma(G, r, \ell) \leq \gamma$, unless $P = NP$.*

The proof of Theorem 5.12 is similar that of Theorem 5.6—in fact, a bit easier, due to the greater flexibility available for defining latency functions. We omit further details.

# 6 Future Work

In this concluding section, we will briefly mention some open questions motivated by our work. While there are recent studies of other network design games [5, 40], we will limit ourselves to the model studied in this paper.

## 6.1 Characterizing Braess's Paradox

In this paper, we have shown that detecting Braess's Paradox is a computationally intractable problem. As discussed in the Introduction, however, this problem seems to become easier when latency functions are not part of the input. To be more precise, call a directed graph $G$ *vulnerable* if it is possible to designate two of its vertices as a source and a destination so that, for some traffic rate $r$ and latency functions $\ell$, there is a subgraph $H$ of $G$ with

$$L(G, r, \ell) > L(H, r, \ell). \tag{8}$$

Vulnerable graphs are therefore the ones susceptible to Braess's Paradox.

**Open Question 1** Characterize the vulnerable graphs.

Milchtaich [76] recently solved Open Question 1 for undirected graphs, thereby confirming an unproven assertion made by Murchland [77].

Replacing the right-hand side of (8) by $c \cdot L(H, r, \ell)$ we obtain the obvious definition of a *c-vulnerable* graph. We urge solvers of Open Question 1 to extend their results in the following way.

**Open Question 2** For all $c \geq 1$, characterize the $c$-vulnerable graphs.

## 6.2 Multi-commodity Flow Networks

In this paper, we have only studied single-commodity networks, where all traffic shares the same source and destination. The traffic model described in this paper extends effortlessly to networks with multiple sources and sinks; details can be found in [93], for example. How severe can Braess's Paradox be in multi-commodity networks? We believe that the following is the most interesting formulation of this question.

**Open Question 3** How much can removing edges from a multi-commodity network decrease the latency of all of the traffic?

To be clear, suppose $(G, r, \ell)$ is a multi-commodity instance. Multi-commodity analogues of Propositions 2.2 and 2.3 enable us to define $L_i(G, r, \ell)$ as the common latency encountered by traffic traveling from the $i$th source to the $i$th destination in a Nash flow for $(G, r, \ell)$. Open Question 3 asks how large the gap

$$\min_i \frac{L_i(G, r, \ell)}{L_i(H, r, \ell)} \tag{9}$$

can be for a subgraph $H$ of a multi-commodity instance $(G, r, \ell)$, as a function of the number of vertices, edges, or commodities of $(G, r, \ell)$. No upper bound and no lower bound better than $n - 1$, where $n$ is the number of vertices of $G$, follow from this work. Open Question 3 was first posed by Weitz [110].

Very recently, Tardos and Walkover [104] proved an upper bound on (9) for multi-commodity networks that is exponential in the number of edges and in the number of commodities. While Lin [71] recently devised new multi-commodity versions of Braess's Paradox, these do not improve upon the lower bound of $n - 1$ for (9).

## 6.3 A General Reduction

In this paper, we considered four different sets of allowable edge latency functions. For each of the four, we proved optimal or near-optimal upper and lower bounds on the approximability of the corresponding network design problem. We used similar proof techniques in the four different cases, and this motivates our final open question.

**Open Question 4** Is there a generic, unifying proof of all of the results in this paper?

To illustrate what we mean in Open Question 4, we will return to Proposition 5.1, where we stated a tight upper bound on how much the cost of a Nash flow can exceed that of an arbitrary feasible flow in a network with polynomial latency functions. Proposition 5.1 is a special case of a much more general result that was proved in [90], which informally states the following: Given *any* set of allowable edge latency functions $\mathcal{L}$, the network with latency functions in $\mathcal{L}$ that is "closest" to Example 4.1 is a worst-possible example. This is a generic reduction from the problem of finding a worst-case network for a set $\mathcal{L}$ of allowable latency functions to what is usually a tractable, back-of-the-envelope calculation. There are also other examples of such generic reductions that are parameterized by the set of allowable edge latency functions [28, 29, 89]. Is there also one that determines a tight bound on the

approximability of the network design problem induced by a given set of allowable latency functions? We expect that a generic inapproximability result will be the biggest obstacle to resolving this question.

## Acknowledgements

We thank Leonard Schulman for introducing us to Braess's Paradox and the Linear Latency Network Design problem, Éva Tardos for helpful discussions and comments on an earlier version of this paper, Dietrich Braess, Hisao Kameda, and Jon Kleinberg for useful conversations, Howard Karloff for bringing [56] to our attention, and the anonymous referees for their comments.

## References

[1] D. Acemoglu and A. Ozdaglar. Flow control, routing, and performance under monopoly pricing. Technical Report WP-1696, MIT LIDS, 2003.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[4] E. Altman, R. El Azouzi, and O. Pourtallier. Avoiding paradoxes in routing games. In *Proceedings of the 17th International Teletraffic Conference*, 2001.

[5] E. Anshelevich, A. Dasgupta, É. Tardos, and T. Wexler. Near-optimal network design with selfish agents. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 511–520, 2003.

[6] R. Arnott, A. De Palma, and R. Lindsey. Properties of dynamic traffic equilibrium involving bottlenecks, including a paradox and metering. *Transportation Science*, 27(2):148–160, 1993.

[7] R. Arnott and K. Small. The economics of traffic congestion. *American Scientist*, 82(5):446–455, 1994.

[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *FOCS '92*.

[9] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *FOCS '92*.

[10] T. Bass. Road to ruin. *Discover*, 13(5):56–61, 1992.

[11] N. Bean. Secrets of network success. *Physics World*, 9(2):30–33, 1996.

[12] N. G. Bean, F. P. Kelly, and P. G. Taylor. Braess's paradox in a loss network. *Journal of Applied Probability*, 34(1):155–159, 1997.

[13] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[14] D. E. Boyce and J. L. Soberanes. Solutions to the optimal network design problem with shipments related to transportation cost. *Transportation Research, Series B*, 13(1):65–80, 1979.

[15] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968. Available from `http://homepage.ruhr-uni-bochum.de/Dietrich.Braess/`.

[16] B. Calvert. The Downs-Thomson effect in a Markov process. *Probability in the Engineering and Informational Sciences*, 11(3):327–340, 1997.

[17] B. Calvert and G. Keady. Braess's paradox and power-law nonlinearities in networks. *Journal of the Australian Mathematical Society, Series B*, 35(1):1–22, 1993.

[18] B. Calvert, W. Solomon, and I. Ziedins. Braess's paradox in a queueing network with state-dependent routing. *Journal of Applied Probability*, 34(1):134–154, 1997.

[19] S. Catoni and S. Pallottino. Traffic equilibrium paradoxes. *Transportation Science*, 25(3):240–244, 1991.

[20] C. K. Chau and K. M. Sim. The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands. *Operations Research Letters*, 31(5):327–335, 2003.

[21] R. Cocchi, S. J. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: Motivation, formulation, and example. *IEEE/ACM Transactions on Networking*, 1(6):614–627, 1993.

[22] J. E. Cohen. The counterintuitive in conflict and cooperation. *American Scientist*, 76(6):577–584, 1988.

[23] J. E. Cohen and P. Horowitz. Paradoxical behavior of mechanical and electrical networks. *Nature*, 352:699–701, August 22, 1991.

[24] J. E. Cohen and C. Jeffries. Congestion resulting from increased capacity in single-server queueing networks. *IEEE/ACM Transactions on Communication*, 5(2):305–310, 1997.

[25] J. E. Cohen and F. P. Kelly. A paradox of congestion in a queuing network. *Journal of Applied Probability*, 27(3):730–734, 1990.

[26] R. Cole, Y. Dodis, and T. Roughgarden. How much can taxes help selfish routing? In *Proceedings of the Fourth Annual ACM Conference on Electronic Commerce*, pages 98–107, 2003.

[27] R. Cole, Y. Dodis, and T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 521–530, 2003.

[28] J. R. Correa, A. S. Schulz, and N. E. Stier Moses. Computational complexity, fairness, and the price of anarchy of the maximum latency problem. Technical Report 4447-03, MIT Sloan Working Paper, 2003.

[29] J. R. Correa, A. S. Schulz, and N. E. Stier Moses. Selfish routing in capacitated networks. Technical Report 4319-03, MIT Sloan Working Paper, 2003. Preliminary version in *SODA '03*.

[30] R. W. Cottle, J. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[31] A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*, pages 287–296, 2002.

[32] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proceedings of the 13th Annual Symposium on Discrete Algorithms*, pages 413–420, 2002.

[33] S. C. Dafermos and A. Nagurney. On some traffic equilibrium theory paradoxes. *Transportation Research, Series B*, 18(2):101–110, 1984.

[34] S. C. Dafermos and F. T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards, Series B*, 73(2):91–118, 1969.

[35] C. F. Daganzo. Queue spillovers in transportation networks with a route choice. *Transportation Science*, 32(1):3–11, 1998.

[36] R. Dionne and M. Florian. Exact and approximate algorithms for optimal network design. *Networks*, 9(1):37–59, 1979.

[37] A. Downs. The law of peak-hour expressway congestion. *Traffic Quarterly*, 16(3):393–409, 1962.

[38] P. Dubey. Inefficiency of Nash equilibria. *Mathematics of Operations Research*, 11(1):1–8, 1986.

[39] R. El Azouzi, E. Altman, and O. Pourtallier. Properties of equilibria in competitive routing with several user types. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pages 3646–3651, 2002.

[40] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. J. Shenker. On a network creation game. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing*, pages 347–351, 2003.

[41] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *FOCS '91*.

[42] C. Fisk. More paradoxes in the equilibrium assignment problem. *Transportation Research, Series B*, 13(4):305–309, 1979.

[43] M. Florian and D. Hearn. Network equilibrium models and algorithms. In M. O. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Routing*, chapter 6, pages 485–550. Elsevier Science, 1995.

[44] S. Fortune, J. E. Hopcroft, and J. C. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[45] M. Frank. The Braess Paradox. *Mathematical Programming*, 20(3):283–302, 1981.

[46] M. Frank. Cost-deceptive links on ladder networks. *Methods of Operations Research*, 45:75–86, 1983.

[47] E. J. Friedman. Genericity and congestion control in selfish routing. Working paper, April 30, 2003. Available from `http://www.orie.cornell.edu/~friedman/papers.htm`.

[48] C. B. Garcia and W. I. Zangwill. *Pathways to Solutions, Fixed Points, and Equilibria.* Prentice-Hall, 1981.

[49] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[50] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 4, pages 144–191. PWS Publishing Company, 1997.

[51] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, 1988. Second corrected edition, 1993.

[52] J. N. Hagstrom and R. A. Abrams. Characterizing Braess's paradox for traffic networks. In *Proceedings of the Fourth IEEE Conference on Intelligent Transportation Systems*, pages 837–842, 2001.

[53] M. A. Hall. Properties of the equilibrium state in transportation networks. *Transportation Science*, 12(3):208–216, 1978.

[54] H. H. Hoc. A computational approach to the selection of an optimal network. *Management Science*, 19(5):488–498, 1973.

[55] A. D. Irvine. How Braess' paradox solves Newcomb's problem. *International Studies in the Philosophy of Science*, 7(2):141–160, 1993.

[56] H. Kameda. How harmful the paradox can be in the Braess/Cohen-Kelly-Jeffries networks. In *Proceedings of 21st INFOCOM Conference*, volume 1, pages 437–445, 2002.

[57] H. Kameda, E. Altman, T. Kozawa, and Y. Hosokawa. Braess-like paradoxes in distributed computer systems. *IEEE Transactions on Automatic Control*, 45(9):1687–1691, 2000.

[58] H. Kameda, E. Altman, J. Li, and Y. Hosokawa. Paradoxes in performance optimization of distributed systems. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, 2000.

[59] G. Keady. The Colebrook-White formula for pipe networks. Electronic report, Department of Mathematics, University of Western Australia, 1995.

[60] F. P. Kelly. Network routing. *Philosophical Transactions of the Royal Society of London, Series A*, 337(1641):343–367, 1991.

[61] S. Khuller. Approximation algorithms for finding highly connected subgraphs. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 6, pages 236–265. PWS Publishing Company, 1997.

[62] F. H. Knight. Some fallacies in the interpretation of social cost. *Quarterly Journal of Economics*, 38(4):582–606, 1924.

[63] G. Kolata. What if they closed 42nd Street and nobody noticed? *New York Times*, page 38, December 25, 1990.

[64] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997.

[65] Y. A. Korilis, A. A. Lazar, and A. Orda. Capacity allocation under noncooperative routing. *IEEE Transactions on Automatic Control*, 42(3):309–325, 1997.

[66] Y. A. Korilis, A. A. Lazar, and A. Orda. Avoiding the Braess paradox in noncooperative networks. *Journal of Applied Probability*, 36(1):211–222, 1999.

[67] T. W. Körner. *The Pleasures of Counting*. Cambridge University Press, 1996.

[68] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[69] L. J. LeBlanc. An algorithm for the discrete network design problem. *Transportation Research*, 9(3):183–199, 1975.

[70] L. Libman and A. Orda. The designer's perspective to atomic noncooperative networks. *IEEE/ACM Transactions on Networking*, 7(6):875–884, 1999.

[71] H. Lin. Personal communication, December 2003.

[72] H. Lin, T. Roughgarden, and É. Tardos. A stronger bound on Braess's Paradox. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 333–334, 2004.

[73] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984.

[74] L. Marinoff. How Braess' paradox solves Newcomb's problem: not! *International Studies in the Philosophy of Science*, 10(3):217–237, 1996.

[75] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 510–519, 2001.

[76] I. Milchtaich. Network topology and the efficiency of equilibrium. Working paper 12-01, Department of Economics, Bar-Ilan University, Israel, 2001.

[77] J. D. Murchland. Braess's paradox of traffic flow. *Transportation Research*, 4(4):391–394, 1970.

[78] A. Nagurney. *Network Economics: A Variational Inequality Approach*. Kluwer, 1993. Revised second edition, 1999.

[79] A. Nagurney. *Sustainable Transportation Networks*. Edward Elgar, 2000.

[80] G. F. Newell. *Traffic Flow on Transportation Networks*. MIT Press, 1980.

[81] C. H. Papadimitriou. Algorithms, games, and the Internet. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 749–753, 2001.

[82] E. I. Pas and S. L. Principio. Braess' paradox: Some new insights. *Transportation Research, Series B*, 31(3):265–276, 1997.

[83] C. M. Penchina. Braess paradox: Maximum penalty in a minimal critical network. *Transportation Research, Series A*, 31(5):379–388, 1997.

[84] I. Peterson. Strings and springs net mechanical surprise. *Science News*, 140(8):118, 1991.

[85] A. C. Pigou. *The economics of welfare*. Macmillan, 1920.

[86] B. Raghavachari. Algorithms for finding low degree structures. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 7, pages 266–295. PWS Publishing Company, 1997.

[87] T. M. Ridley. An investment policy to reduce the travel time in a transportation network. *Transportation Research*, 2(4):409–424, 1968.

[88] T. Roughgarden. Stackelberg scheduling strategies. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 104–113, 2001. To appear in *SIAM Journal of Computing*.

[89] T. Roughgarden. How unfair is optimal routing? In *Proceedings of the 13th Annual Symposium on Discrete Algorithms*, pages 203–204, 2002.

[90] T. Roughgarden. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67(2):341–364, 2003. Preliminary version in *STOC '02*.

[91] T. Roughgarden. The maximum latency of selfish routing. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, pages 973–974, 2004.

[92] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, 2004 (forthcoming).

[93] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002. Preliminary version in *FOCS '00*.

[94] T. Roughgarden and É. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behavior*, 2004. To appear.

[95] L. Schulman. Personal communication, October 1999.

[96] A. J. Scott. The optimal network problem: Some computational procedures. *Transportation Research*, 3(2):201–210, 1969.

[97] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, 1985.

[98] S. J. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEE/ACM Transactions on Networking*, 3(6):819–831, 1995.

[99] M. J. Smith. In a road network, increasing delay locally can reduce delay globally. *Transportation Research*, 12(6):419–422, 1978.

[100] R. Steinberg and R. E. Stone. The prevalence of paradoxes in transportation equilibrium problems. *Transportation Science*, 22(4):231–241, 1988.

[101] R. Steinberg and W. I. Zangwill. The prevalence of Braess' Paradox. *Transportation Science*, 17(3):301–318, 1983.

[102] P. D. Straffin. *Game Theory and Strategy.* Mathematical Association of America, 1993.

[103] A. Taguchi. Braess' paradox in a two-terminal transportation network. *Journal of the Operations Research Society of Japan*, 25(4):376–388, 1982.

[104] É. Tardos and A. Walkover. The maximum latency of multicommodity selfish routing. In preparation, 2004.

[105] R. E. Tarjan. *Data Structures and Network Algorithms.* SIAM, 1983.

[106] J. M. Thomson. *Great Cities and Their Traffic.* Gollancz, 1977.

[107] A. Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science*, pages 416–425, 2002.

[108] W. S. Vickrey. Congestion theory and transport investment. *American Economic Review*, 59(2):251–260, 1969.

[109] J. G. Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institute of Civil Engineers, Pt. II*, volume 1, pages 325–378, 1952.

[110] D. Weitz. The price of anarchy. Unpublished manuscript, 2001.

[111] W. Whitt. Counterexamples for comparisons of queues with finite waiting rooms. *Queueing Systems*, 10(3):271–278, 1992.

[112] R. T. Wong. Introduction and recent advances in network design: Models and algorithms. In M. Florian, editor, *Transportation Planning Models*, pages 187–225. Elsevier Science, 1984.

[113] H. Yang and M. G. H. Bell. A capacity paradox in network design and how to avoid it. *Transportation Research, Series A*, 32(7):539–545, 1998.