Approximating Multi Commodity Network Design on Graphs of Bounded Pathwidth and Bounded Degree

Kord Eickmeyer* and Ken-ichi Kawarabayashi

National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
{eickmeye,k_keniti}@nii.ac.jp

Abstract. In the *Multicommodity Network Design* problem (MCND) we are given a digraph G together with latency functions on its edges and specified flow requests between certain pairs of vertices. A flow satisfying these requests is said to be at Nash equilibrium if every path which carries a positive amount of flow is a shortest path between its source and sink. The goal of MCND is to find a subgraph H of G such that the flow at Nash equilibrium in H is optimal. While this has been shown to be hard to approximate (with multiplicative error) for a fairly large class of graphs and latency functions, we present an algorithm which computes solutions with small additive error in polynomial time, assuming the graph G is of bounded degree and bounded path-width, and the latency functions are Lipschitz-continuous. Previous hardness results in particular apply to graphs of bounded degree and graphs of bounded path-width, so it is not possible to drop one of these assumptions.

1 Introduction

We model road networks by directed graphs whose edges are labelled with *latency functions*, i.e., functions which express the expected time it takes to traverse the edge depending on the amount of traffic taking it. Adding the assumption that each driver will take a route which, given the current traffic situation, has shortest travel time, one arrives at the model of *selfish routing*, which we review in Section 1.1. Surprisingly, simple examples show that, in this model, removing edges from the network may improve the perfomance of the network, in the sense that the travel time of all participants may be reduced. This phenomenon is called *Braess's paradox* after Dietrich Braess, who first described it in [1].

The obvious question of which edges should be removed to yield an optimal traffic situation is called Multicommodity Network Design Problem (MCND) and has been shown to be computationally hard to solve even approximately, see section 1.2 for details. Our main contribution is a polynomial time approximation algorithm on inputs in which

* This work was supported by a fellowship of the first author within the FIT-Programme of the German Academic Exchange Service (DAAD).

– the input graph is of bounded path-width and bounded degree and
– the time it takes to traverse an edges depends in a Lipschitz continuous way on the amount of traffic traversing that edge.

The algorithm returns a subgraph in which an ϵ-approximate Nash equilibrium exists which is at most by an additive term of ϵ worse than the best γ-Nash equilibrium in any other subgraph, for some γ depending on ϵ and which is smaller than ϵ, see Definition 6. Here, an ϵ-Nash equilibrium is a flow in which all traffic is routed along paths which are at most an additive term of ϵ worse than shortest paths. The proof is contained in Theorem 7 and Lemma 10.

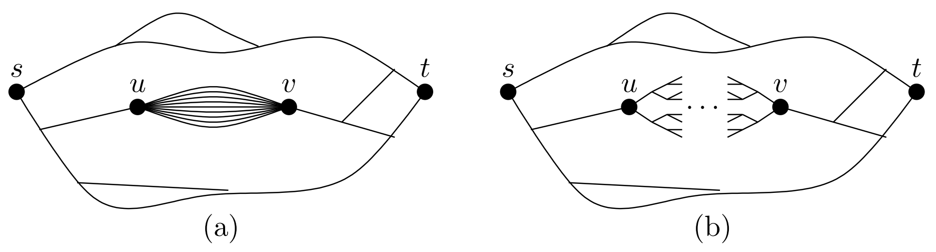

Fig. 1. (a) The reduction in [2] produces very simple graphs in which all of the input is encoded in a set of parallel edges between a certain pair of vertices (u, v). (b) By replacing the parallel edges with binary trees one obtains graphs of bounded degree.

Assuming the input graph to be simultaneously of bounded degree and bounded path-width is a strong restriction. However, previous hardness results (B2, B3 of Section 1.2) showed that MCND is hard even on very simple graphs (planar, acyclic) with just one source of complexity: There are pairs of nodes with many disjoint paths between them. If these paths are short (or even parallel edges), then there must be vertices of high degree (cf. Fig 1a). If we bound the maximum degree there may still be pairs of vertices with many paths between them, by replacing nodes of high degree with binary trees (cf. Fig 1b). We rule this out by bounding the path-width. Note that our approximation algorithm works with approximate Nash equilibria, so technically the hardness results mentioned here do not apply exactly. However, they can be adapted to include approximate Nash equilibria.

Previous attempts at obtaining approximation algorithms for the (single commodity) network design problem on restricted instances include Fotakis et al. [3]. Using a probabilistic argument, they show that *approximate* Nash equilibria can be found in a restricted search space, which yields a polynomial time algorithm on instances with only polynomially many paths from the source to the sink whose length is polylogarithmical in the number of edges of the graph.

1.1 Selfish Routing

We follow the definitions and notation of [4]. Let $G = (V, E)$ be a digraph, and let (s_i, t_i) be k pairs of designated vertices. Furthermore, for each edge $e \in E$ there is an associated *latency function* $l_e : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and for each pair (s_i, t_i) we are given a flow request $r_i \in \mathbb{R}_{\geq 0}$. We assume all latency functions to be continuous and non-decreasing. We frequently denote the number of vertices by n and the number of edges by m. We use the notation $[l] := \{1, \ldots, l\}$ for every natural number l.

The intuition behind the problem is that we want to route r_i units of traffic from each of the source nodes s_i to the corresponding target node t_i. The traffic is supposed to consist of infinitesimally small pieces, so that it may be split arbitrarily among the possible paths connecting s_i to t_i in the graph. Formally, by a u-v-path P we mean a sequence $u = v_0, v_1, \ldots, v_n = v$ of pairwise distinct nodes $v_i \in V$ such that $(v_{i-1}, v_i) \in E$ for all $i \in [n]$. Denoting the set of all s_i-t_i-paths in G by $\mathcal{P}_i$ and the set of all paths between pairs of vertices in the problem by $\mathcal{P} = \cup_i \mathcal{P}_i$, a *flow f feasible for* (G, r, l) is any function $f : \mathcal{P} \to \mathbb{R}_{\geq 0}$ such that for each $i \in [k]$ the equation $\sum_{P \in \mathcal{P}_i} f_P = r_i$ is satisfied. We denote the set of all paths in the graph by $\mathcal{P}_{\text{all}} := \bigcup_{u,v \in V} \{P \mid P \text{ is a } u\text{-}v\text{-path}\}$. The *length* $|P|$ of a path $P \in \mathcal{P}_{\text{all}}$ is the number of edges in P.

The latency function l_e specifies the amount of time it takes to traverse the edge e, as a function of the amount of traffic being routed along this edge. More traffic might cause congestion and therefore increase this time. The flow function f can be seen as a way of assigning a path $P \in \mathcal{P}_i$ to each of the infinitesimal atoms (say, cars) composing the traffic from s_i to t_i.

For an edge $e \in E$ and a flow f, denote by f_e the total amount of traffic that is routed along e, i.e., $f_e = \sum_{P \in \mathcal{P}, e \in P} f_P$. The *latency $l_P(f)$ of a path $P \in \mathcal{P}_{\text{all}}$* given a specific flow f is the sum of the latencies of all edges along this path, i.e., $l_P(f) = \sum_{e \in P} l_e(f_e)$.

The assumption that each atom behaves selfishly is captured in the notion of Nash equilibrium: A feasible flow f is said to be at Nash equilibrium if

$$l_P(f) \leq l_{P'}(f) \quad \text{for all } i \in [k] \text{ and all } P, P' \in \mathcal{P}_i \text{ with } f_P > 0.$$

Note that since the atoms are infinitesimally small, a single deviation will not change the latencies $l_e(f_e)$.[1]

1.2 Braess's Paradox

By the definition of Nash equilibrium, all s_i-t_i-paths $P \in \mathcal{P}_i$ actually carrying flow (i.e., $f_P > 0$) must have the same latency, which we denote by $L_i(f)$. It can be shown ([4, Cor. 2.6.2]) that for an instance (G, r, l), all Nash equilibria yield the same edge latencies, i.e., $l_e(f_e) = l_e({f'}_e)$ for all edges $e \in E$ and flows f, f'

[1] Alternatively, taking some $\delta > 0$ as atomic amount of traffic and letting δ tend to zero yields the same notion of Nash equilibrium.

both at Nash equilibrium. In particular the maximum latency

$$M(f) := \max_i L_i(f)$$

of a Nash equilibrium of an instance (G, r, l) is well-defined for each instance; we denote this by $M(G, r, l)$.

Braess's paradox amounts to the fact that there may be a subgraph $H \leq G$, obtained from G by removing edges, such that $M(H, r, l) < M(G, r, l)$. We define the *Braess ratio* of (G, r, l) as

$$B(G, r, l) := \max_{H \leq G} \frac{M(G, r, l)}{M(H, r, l)},$$

with the convention that $0/0 := 1$; if $M(H, r, l) = 0$ for some $H \leq G$ then also $M(G, r, l) = 0$. Braess's paradox has been studied intensively recently, and there are strong bounds on the Braess ratio:

(A1) In *single-commodity instances* (G, r, l), i.e., when $k = 1$, then $B(G, r, l) \leq \lfloor \frac{n}{2} \rfloor$, and this bound is optimal [5].[2] Moreover [2], if every matching of $V \setminus \{s, t\}$ using only edges in $G \setminus H$ has size at most c, then

$$L(G, r, l) \leq (c + 1)L(H, r, l).$$

In particular, removing c edges from G may only reduce $L(G, r, l)$ by a factor of $1/(c + 1)$.

(A2) In multi-commodity instances, there are (G, r, l) with $B(G, r, l) = 2^{\Omega(n)}$, and this ratio may be attained by removing a single edge from G (see [2]). The same paper contains an upper bound of $2^{O(\min\{m \log n, kn\})}$ on the Braess ratio in arbitrary networks.

(A3) In single-commodity instances with linear latency functions (i.e., $l_e(x) = a_e + b_e \cdot x$ for all $e \in E$), the Braess ratio is at most 4/3. Again, this bound may actually be attained by removing a single edge (see [4]).

(A4) In [5], Roughgarden defines the *incline* $\Gamma(c)$ of a continuous monotonely increasing function $c : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as $\Gamma(c) := \sup_{x>0} \frac{x \cdot c(x)}{\int_0^x c(t)\,\mathrm{d}t}$, with $0/0 := 1$. With this definition, he obtains the bound $B(G, r, l) \leq \gamma$ for all instances (G, r, l) with $\Gamma(l_e) \leq \gamma$ for all $e \in E$.

Finding a subgraph $H \leq G$ which minimises $M(H, r, l)$ is a natural algorithmic question. This problem, called *Multicommodity Network Design (MCND)* is hard to approximate in the following sense: For $\delta > 1$, we say that an algorithm is a δ-approximation algorithm for MCND if it computes, on input (G, r, l), a subgraph $H \leq G$ with

$$M(H, r, l) \leq \delta \cdot \min_{H' \leq G} M(H', r, l).$$

Assuming P $\neq$ NP,

[2] Note that, in single-commodity instances, $M(G, r, l) = L(G, r, l)$.

(B1) there is no polynomial-time $(\frac{4}{3} - \epsilon)$-approximation algorithm for single-commodity network design with linear latency functions [5], for any $\epsilon > 0$,

(B2) there is no polynomial-time $(n/2 - \epsilon)$-approximation algorithm for single-commodity network design with arbitrary latency functions [5], for any $\epsilon > 0$,

(B3) there is no polynomial-time $2^{o(n)}$-approximation algorithm for multi-commodity network design [2].

(B4) 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, l) with $\Gamma(l_e) \leq \gamma$ for all $e \in E$ [5].

Note that A1 and A3 imply that in the first two cases, the trivial algorithm (which always returns the whole graph G) is a best-possible approximation algorithm.

These results are proved by reducing NP-complete problems to the appropriate network design problems. For B2 and B3, the reductions produce acyclic planar graphs which contain pairs of vertices with many parallel edges (Fig. 1a). The latency functions used in these reductions are continuous approximations of step functions and therefore increase very steeply in very small regions.

Since step functions can not be approximated sufficiently well (for the purpose of the above reductions) by linear functions or functions with bounded incline, the proofs for B1 and B4 use an entirely different approach. Here, the reductions start from the problem 2-Directed Vertex Disjoint Paths (2DDP) of finding, in a directed graph with two designated pairs (s_1, t_1) and (s_2, t_2) of vertices, a pair of vertex-disjoint paths between them. This problem has been shown to be NP-hard for general graphs [6], but it is solvable in polynomial time on planar graphs [7]. On the other hand, by inspection we see that the reduction B1 actually uses only latency functions with slopes 0 and 1.

2 An Approximation Algorithm for MCND

In this section we will describe an approximation algorithm for MCND on a restricted set of instances. The restriction is two-fold: We impose restrictions on the graph G (see Definition 1 and section 3), and we bound the slope of the admissible latency functions.

In contrast to the inapproximability results mentioned in section 1, we obtain approximations up to an *additive*, rather than multiplicative, error. Because in all of the non-approximability results of section 1 there was no s_i-t_i-path of latency less than 1, these results also hold for additive errors (note that $c + \epsilon < c \cdot (1 + \epsilon)$ if $c \geq 1$).

We discretise the MCND problem in two ways:

1. We consider only flows which route integer multiples of γ along each edge, for some $\gamma > 0$ and
2. we approximate path latencies up to an additive error of γ.

With these discretisations, we are able to approximately solve MCND on instances for which the latency functions are Lipschitz continuous (see below) and for which the underlying graph is of bounded total degree and bounded path-width (ignoring the edge directions). For such graphs it is possible to compute the following kind of decomposition:

Definition 1. *Let $G = (V, E)$ be a directed graph. A* strong directed path-decomposition (sdpd) *is a sequence of subsets $\emptyset = A_0, \ldots, A_m \subset V$ (called* bags*) such that:*

(SDPD1) $|A_j \setminus A_{j-1}| \leq 1$ *for all $j \in [m]$, and if $v \in A_j \setminus A_{j-1}$, then all incoming edges to v come from vertices in A_{j-1},*

(SDPD2) for every $v \in V$ there are $1 \leq a \leq b \leq m$ such that

$$v \in A_j \quad \Leftrightarrow \quad a \leq j \leq b$$

for all $j = 0, \ldots, m$ (in particular, $\bigcup_j A_j = V$).

For a vertex $v \in V$ we denote by $\iota(v) := \min\{j \mid v \in A_j\}$ the index of the first bag which contains it. We call m the length *and $\max_i |A_i| - 1$ the* width *of the decomposition. We define $G_j := (V_j, E_j)$ to be the subgraph of G induced on the set of vertices in the bags up to A_j, i.e.,*

$$V_j := \bigcup_{j' \leq j} A_{j'} \qquad \text{and} \qquad E_j := E \cap (V_j \times V_j).$$

We use the term *strong* to distinguish our definition from the definition of a directed path-decomposition given, e.g., by Barát in [8], where (SDPD1) is replaced by the weaker condition that for all edges uv there be indices $i \leq j$ with $u \in A_i$ and $v \in A_j$. In particular, any acyclic digraph has directed path-width 0, but the strong directed path-width may be arbitrarily high. Also, to simplify the presentation we require that $|A_j \setminus A_{j-1}|$ be at most 1, but it should be clear how to adjust the update step of Theorem 7 to the case $|A_j \setminus A_{j-1}| \geq 2$. In Section 3 we show how to find sdpds in polynomial time in graphs of bounded path-width and bounded degree.

The important feature of sdpds which we use in our algorithm is that

1. every bag A_j separates G_j from $\bar{G_j} := G \setminus G_j$ and
2. all edges between G_j and $\bar{G_j}$ are directed from the former to the latter.

Consequently, any path between vertices $u, v \in V_j$ stays entirely in G_j. Directed path-decompositions in the sense of [8] only have the second property, while undirected path-decompositions (see Definition 8) have only the first one.

In general, discretising the amount of traffic that may travel along each edge might drastically change the set of Nash equilibria. In order to prevent this, we use the following continuity condition:

Definition 2. *For $\alpha > 0$ we call an instance (G, r, l) of MCND* α-Lipschitz continuous *if all latency functions are Lipschitz continuous with Lipschitz constant α, i.e.,*

$$|l_e(x) - l_e(y)| \leq \alpha\, |x - y|$$

for all edges $e \in E$ and $x, y \in \mathbb{R}_{\geq 0}$.

Lipschitz continuity ensures that we may slightly change a flow without changing path latencies by too much:

Lemma 3. *Let $\epsilon, \alpha > 0$ and let (G, r, l) be an α-Lipschitz continuous instance of MCND. If f and f' are flows for which $|f_e - f'_e| < \epsilon$ for all edges e, and $P \in \mathcal{P}_{\text{all}}$ is a path, then*

$$|l_P(f) - l_P(f')| < |P|\, \alpha\epsilon$$

We omit the straightforward proof. Putting these two together we obtain the following lemma for discretised flows:

Lemma 4. *Let $\alpha > 0$ and let (G, r, l) be an α-Lipschitz continuous instance of MCND, and $A_0, \ldots, A_m$ is an sdpd of G. Let $\gamma > 0$ and assume that all r_i are integer multiples of γ. Then if $f : \mathcal{P} \to \mathbb{R}_{\geq 0}$ is a flow in (G, r, l), there is a flow f' such that*

- *f'_e is an integer multiple of γ for all edges $e \in E$ and*
- *$|f_e - f'_e| < \iota(v)\gamma$ for all $e \in E$ directed towards the node $v \in V$.*
- *$|l_P(f) - l_P(f')| < \alpha\gamma\, |P|\, \iota(v)$ for all simple paths $P \in \mathcal{P}_{\text{all}}$ ending in a vertex $v \in V$.*

Proof. We can obtain the flow f' by discretising along the decomposition, maintaining the conditions that the total flow which f' routes into vertices in A_j is an integer multiple of γ and does not differ from the flow which f routes into these vertices by more than $j\gamma$. The statement about path latencies is an easy consequence of Lemma 3. □

We relax the notion of a Nash equilibrium as follows:

Definition 5. *Let (G, r, l) be an instance of selfish routing and $\epsilon > 0$. An* ϵ-Nash equilibrium *is a flow f for (G, r, l) such that for all i and all paths $P, P' \in \mathcal{P}_i$ we have*

$$l_P(f) \leq l_{P'}(f) + \epsilon \quad \text{if } f_P > 0.$$

Accordingly, we define

$$M_{\epsilon\text{-}Nash}(G, r, l) := \min\{M(f) \mid f \text{ is } \epsilon\text{-Nash equilibrium}\},$$

where $M(f)$ is the maximum latency of a path from some s_i to some t_i in the flow f.

Note that we use an additive error here, as opposed to a factor of $(1 + \epsilon)$ as in Roughgarden's definition of ϵ-approximate Nash equilibria [4, Sec. 4.2]. We can now make precise what we mean by "approximately solving MCND":

Definition 6. *Let (G, r, l) be an instance of MCND, and $\epsilon, \gamma > 0$. An (ϵ, γ)-*approximate solution to MCND *is a subgraph $H \subseteq G$ such that*

$$M_{\epsilon\text{-}Nash}(H, r, l) \leq M_{\gamma-Nash}(H', r, l) + \epsilon$$

for all subgraphs $H' \subseteq G$.

We will invoke this definition with γ much smaller than ϵ. With these definitions we are ready to state our main approximation result:

Theorem 7. *Let $\alpha, \epsilon > 0$ and $k, \rho \in \mathbb{N}$ be fixed. Assume we are given an α-Lipschitz continuous MCND instance (G, r, l) with k source-sink pairs $s_1, t_1, \ldots, s_k, t_k$, together with a strong directed path-decomposition $A_0, \ldots, A_m \subset V$ of width ρ. Then it is possible to find an (ϵ, γ)-approximate solution to MCND in time polynomial in the size of G and $\max r_i$, for $\gamma := \frac{\epsilon}{1+\alpha m^2}$.*

Proof. Let M be an upper bound on all flow requests r_i and on the latency of an s_i-t_i-path in a Nash equilibrium in the input graph G, and denote by k the number of source-sink pairs in (G, r, l). For a fixed Lipschitz constant α, this bound can be taken to be $\alpha |V| \max r_i$.

We discretise flows and approximate latencies with the γ stated in the theorem. For each A_j we compute a table T_j which is indexed by all tuples $(\sigma_j^{(1)}, \ldots, \sigma_j^{(k)}, \lambda_j^{(1)}, \ldots, \lambda_j^{(k)})$ of functions such that

– $\sigma_j^{(i)}$ is a function from A_j to $\mathbb{R}_{\geq 0} \cup \{\perp, \top\}$ such that if $\sigma_j^{(i)}(A_j) \subset \mathbb{R}_{\geq 0}$ then

$$\sum_{v \in A_j} \sigma_j^{(i)}(v) = r_i$$

  and $\sigma_j^{(i)}(v)$ is an integer multiple of γ for all $v \in A_j$. If $\sigma_j^{(i)}(v) \in \{\perp, \top\}$ for some $v \in A_j$ we demand $\sigma_j^{(i)}(w) = \sigma_j^{(i)}(v)$ for all $w \in A_j$.

– $\lambda_j^{(i)}$ is a function from A_j to $[0, M]$ such that every $\lambda_j^{(i)}(v)$ is an integer multiple of γ for all $v \in A_j$.

An index $(\sigma_j^{(1)}, \ldots, \sigma_j^{(k)}, \lambda_j^{(1)}, \ldots, \lambda_j^{(k)})$ is meant to represent a flow routing approximately $\sigma_j^{(i)}(v)$ of traffic from s_i to v, such that the common latency of all s_i-v-paths is roughly $\lambda_j^{(i)}(v)$. The special symbol $\perp$ signifies that $s_i \notin G_j$, while $\top$ signifies that $t_i \in G_j$. Note that the size of this index set is linear in k and polynomial of degree ρ in M for fixed values of ρ and ϵ (note that $m < M$). An entry in the table may be $\perp$ or a subset of the edges of G_j. We define

$$B_j := 1 + \alpha j^2$$

The table entries will satisfy the following conditions:

(a) If $T_j(\sigma_j^{(1)}, \dots, \sigma_j^{(k)}, \lambda_j^{(1)}, \dots, \lambda_j^{(k)})$ is a subset of E, then after removing these edges there is a flow f in G_j which routes an amount $\sigma_j^{(i)}(v)$ of traffic from source s_i to $v \in A_j$, such that for each path P from s_i to v with $f_P > 0$ the latency $l_P(f)$ is within $B_j\gamma$ of $\lambda_j^{(i)}(v)$, and such that the flow f is a $B_j\gamma$-Nash equilibrium. If $s_i \notin V_j$ then we demand $\sigma_j^{(i)} \equiv \perp$, and if $t_i \in G_j$ we demand $\sigma_j^{(i)} \equiv \top$.

(b) If $T_j(\sigma_j^{(1)}, \dots, \sigma_j^{(k)}, \lambda_j^{(1)}, \dots, \lambda_j^{(k)}) = \perp$, then no way of removing edges from G will yield the existence of a flow f routing $\sigma_j^{(i)}(v)$ traffic from source s_i to v in such a way that f is a γ-Nash equilibrium and such that the latency from source s_i to v under this flow is $\lambda_j^{(i)}(v)$.

We successively compute the entries of T_j as follows:

– For T_0 we set all entries to $\perp$ except for the one corresponding to $\sigma_0^{(i)} \equiv \perp$ and $\lambda_0^{(i)} \equiv 0$ for all $i \in [k]$, which we set to $\emptyset$.

– Let $A_j \setminus A_{j-1} = \{v\}$ for some node $v \notin \{s_1, \dots, s_k, t_1, \dots, t_k\}$, and let $U = \{u_1, \dots, u_h\} \subseteq A_{j-1}$ be the starting points of all incoming edges to v. Let $\sigma_j^{(1)}, \dots, \sigma_j^{(k)}, \lambda_j^{(1)}, \dots, \lambda_j^{(k)}$ be an index into the table T_j.
If $s_i \notin G_j$ we ignore commodity i in the following discussion and focus on indices (both into T_{j-1} and into T_j) with $\sigma_j^{(i)} \equiv \perp$ and $\lambda_j^{(i)} \equiv 0$. Similarly, if $t_i \in G_{j-1}$ we focus on indices with $\sigma_j^{(i)} \equiv \top$ and $\lambda_j^{(i)} \equiv 0$.
To determine the entry $T_j(\sigma_j^{(1)}, \dots, \sigma_j^{(k)}, \lambda_j^{(1)}, \dots, \lambda_j^{(k)})$, we use the values in the table T_{j-1} to determine possible ways of routing traffic to the nodes in A_{j-1}, and try to extend these flows to a flow which is still an approximate Nash equilibrium and such that the new flow routes $\sigma_j^{(i)}$ of commodity i to the nodes in A_j, and such that the latency of travelling from source s_i to $v \in A_j$ is $\lambda_j^{(i)}(v)$, up to an additive error of $j\epsilon/m$. We need only change flows to A_{j-1} in such a way that we additionally route traffic from the nodes $u_1, \dots, u_h \in A_{j-1}$ to v, as re-routing traffic between the nodes in A_{j-1} only results in flows which have already been considered when computing the table T_{j-1}.
We are looking for an index $(\sigma_{j-1}^{(1)}, \dots, \sigma_{j-1}^{(k)}, \lambda_{j-1}^{(1)}, \dots, \lambda_{j-1}^{(k)})$ into the table T_{j-1}, a subset $S \subset \{u_1, \dots, u_h\}$ of predecessors of v and real numbers $\phi_{i,w} \in \mathbb{R}_{\geq 0}$ for $w \in S$ and $i \in [k]$ such that:
  - There is a way of removing edges from G_{j-1} to yield the existence of an approximate Nash equilibrium up to A_{j-1}, i.e.,

$$T_{j-1}(\sigma_{j-1}^{(1)}, \dots, \sigma_{j-1}^{(k)}, \lambda_{j-1}^{(1)}, \dots, \lambda_{j-1}^{(k)}) \neq \perp.$$

  - Routing $\phi_{i,w}$ of commodity i from w to v changes $\sigma_{j-1}^{(i)}$ into $\sigma_j^{(i)}$:

$$\sum_{w \in S} \phi_{i,w} = \sigma_j^{(i)}(v) \quad \text{for all } i = 1, \dots, k$$

and for all $w \in S$ and $i = 1, \dots, k$ we have

$$\sigma_{j-1}^{(i)}(w) - \phi_{i,w} = \begin{cases} \sigma_j^{(i)}(w) & \text{if } w \in A_j \\ 0 & \text{otherwise} \end{cases}$$

In particular, the $\phi_{i,w}$ are also integer multiples of ϵ/m.

- If $\sigma_j^{(i)}(v) = 0$ then also $\lambda_j^{(i)}(v) = 0$. Otherwise, the latencies of s_i-v-paths are approximately given by $\lambda_j^{(i)}(v)$: If $\phi_{i,w} > 0$, then

$$\left| \lambda_{j-1}^{(i)}(w) + l_{(wv)} \left(\sum_{i=1}^{k} \phi_{i,w} \right) - \lambda_j^{(i)}(v) \right| \leq (B_j - B_{j-1})\gamma$$

  and the flow is still approximately at Nash equilibirium: For all $w \in S$ and $i \in [k]$ with $\sigma_{j-1}^{(i)}(w) > 0$,

$$\lambda_{j-1}^{(i)}(w) + l_{(wv)} \left(\sum_{i=1}^{k} \phi_{i,w} \right) \geq \lambda_j^{(i)}(v) - \frac{\epsilon}{2}.$$

- The approximate latencies for vertices in $A_j \cap A_{j-1}$ remain unchanged unless $\sigma_j^{(i)}(v) = 0$: $\lambda_j^{(i)}(u) = \lambda_{j-1}^{(i)}(u)$ for all $i \in [k]$ and $u \in A_j \cup A_{j-1}$.

If there is such a combination, then we set

$$T_j(\sigma_j^{(1)}, \dots, \sigma_j^{(k)}, \lambda_j^{(1)}, \dots, \lambda_j^{(k)}) := T_{j-1}(\sigma_{j-1}^{(1)}, \dots, \sigma_{j-1}^{(k)}, \lambda_{j-1}^{(1)}, \dots, \lambda_{j-1}^{(k)}) \\ \cup \{uv \mid u \in U \setminus S\}$$

– If $A_j \setminus A_{j-1} = \{s_i\}$ for a source node s_i we proceed essentially as in the previous step but demand that $\sigma_j^{(i)}(v) = r_i$ and $\lambda_j^{(i)} = 0$. The other latencies and flows of other commodities into s_i are handled as above. Note that this can be adjusted to the case where $s_i = s_{i'}$ for $i \neq i'$.

– Finally, if $A_j \setminus A_{j-1} = \{t_i\}$ for a source node t_i we demand that $\sigma_j^{(i)}(v) = \top$ treat this to mean that exactly an amount r_i of commodity enters t_i. The other latencies and flows of other commodities into t_i are handled as above.

That this way of filling the table will satisfy condition (a) is easily verified. We now turn to condition (b). Assume that for some bag A_j and some index $(\sigma, \lambda) = (\sigma_j^{(1)}, \dots, \sigma_j^{(k)}, \lambda_j^{(1)}, \dots, \lambda_j^{(k)})$ into T_j we have $T_j(\sigma, \lambda) = \perp$ but still there is a subset $F \subset E_j$ and a flow f the graph G_j with the edges in F removed such that f is a γ-Nash equilibrium which routes $\lambda_j^{(i)}(v)$ of commodity i into v.

Using Lemma 4, we discretise f to obtain a flow f' such that $|l_P(f) - l_P(f')| < \alpha j^2 \gamma$ for all paths $P \in \mathcal{P}_{\text{all}}$ with endpoints in A_j. Since we assumed f to be a γ-Nash equilibrium, for two s_i-u-paths P and P' with endpoint $u \in A_j$ we have

$$\begin{aligned} |l_P(f') - l_{P'}(f')| &\leq |l_P(f') - l_P(f)| + |l_P(f) - l_{P'}(f)| + |l_{P'}(f) - l_{P'}(f')| \\ &\leq \alpha\gamma j^2 + \gamma + \alpha\gamma j^2 \quad \leq \quad (2B_j - 1)\gamma \end{aligned}$$

In particular, there is an integer multiple λ of γ such that all latencies $l_P(f')$ for s_i-u paths p are within distance B_j of λ. Following the computation path we see that the corresponding table entry in T_j can not be $\perp$. □

3 Graphs of Bounded Path-Width

While the strong directed path-decompositions of Definition 1 are convenient for the purpose of our algorithm, they are non-standard and it is not clear what kinds of graphs allow for these decompositions and how they can be obtained. In this section we show that in particular graphs of bounded path-width and simultaneously bounded degree allow for such decompositions.

Path-width was defined by Robertson and Seymour in the first paper of their Graph Minors series [9]:

Definition 8. *Let $G = (V, E)$ be an undirected graph. A* path-decomposition *of G is a sequence $X_1, \ldots, X_s$ of subsets of V such that*

- *for every $e \in E$ there is an $i \in [s]$ such that both endpoints of e are in X_i and*
- *$X_i \cap X_k \subseteq X_j$ for all $1 \leq i \leq j \leq k \leq s$.*

The width *of the decomposition is $\max_i |X_i| - 1$. The* path-width *ρ of G is the minimum $\rho \in \mathbb{N}$ such that G has a path-decomposition of width ρ.*

We will need the following fact about graphs of bounded path-width:

Fact 9. *For every $\rho \in \mathbb{N}$ there is an $h \in \mathbb{N}$ such that no graph of path-width at most ρ has a minor isomorphic to the complete binary tree of height h.*

This follows easily from theorem (1.2) in [9] and the fact that every tree is a minor of a sufficiently large complete binary tree. As usual, a minor of a graph G is a graph obtained from a subgraph of G by contracting edges.

We are now ready to state the main result of this section:

Lemma 10. *Let $b \in \mathbb{N}$ and let $\mathcal{G}$ be a class of acyclic directed graphs such that for every $G \in \mathcal{G}$, the total degree (i.e., in-degree plus out-degree) of every vertex $v \in V(G)$ is at most b and the (undirected) path-width of G is at most b. Then there is a $\rho \in \mathbb{N}$ depending only on b such that, given a graph $G \in \mathcal{G}$, an sdpd of G of width at most ρ can be computed from G in linear time.*

Proof (Proof of Lemma 10). Using Fact 9, let $d_0 \in \mathbb{N}$ be such that none of the graphs in $\mathcal{G}$ contains a complete binary tree of depth d_0 as a minor (ignoring all edge directions).

Pick some $G \in \mathcal{G}$. We may assume that G has exactly one sink. Otherwise, let $X_1, \ldots, X_l$ be a path-decomposition of G, and let $v_1, \ldots, v_s$ be the sinks of G in the order in which they first appear in the path-decomposition, breaking ties arbitrarily. Adding a directed edge from v_i to v_{i+1} for $i \in [s-1]$ increases the path-width of G by at most 1, because we can obtain a path-decomposition for the new graph by adding v_i to all bags between the first entry of v_i and the first entry of v_{i+1}, increasing each bag-size by at most one. Furthermore, we only increase the degree of the graph by at most two. Acyclicity is also maintained, and by a result of Bodlaender and Kloks [10], for fixed path-width b a path-decomposition of width b can be computed in linear time.

To obtain an sdpd for G, we start from the sink and successively create new bags by taking all predecessors of all nodes in the current bag. If, at some point, the resulting bag has size exceeding b^{d_0}, then G has a minor isomorphic to a complete binary tree of depth d_0, a contradiction. $\square$

Conclusion

We complemented Roughgarden's [5] and Lin et al.'s [2] results on the hardness of approximation (up to a multiplicative error) of the multi-commodity network design problem by giving an approximation algorithm for this problem on a certain restricted class of inputs, namely graphs allowing for what we call a bounded path-decomposition with Lipschitz-continuous latency functions. For technical reasons, we have to work with approximate Nash equilibria, so our algorithm does not directly compare with previous hardness results.

For general latency functions, restrictions on the class of input graphs similar to ours seem to be necessary [2]. If the latency functions are polynomials of bounded degree, the proof technique used in [5] combined with Schrijver's algorithm for 2DDP on planar graphs [7] raises the question of whether efficient approximation algorithms exist for less severely restricted classes of input graphs such as planar graphs.

References

1. Braess, D.: Über ein Paradoxon aus der Verkehrsplanung. Mathematical Methods of Operations Research **12**(1) (1968) 258–268
2. Lin, H.C., Roughgarden, T., Tardos, É., Walkover, A.: Stronger bounds on Braess's paradox and the maximum latency of selfish routing. SIAM J. Discrete Math. **25**(4) (2011) 1667–1686
3. Fotakis, D., Kaporis, A.C., Spirakis, P.G.: Efficient methods for selfish network design. In: ICALP Track C. Volume 5556 of LNCS. (2009) 459–471
4. Roughgarden, T.: Selfish Routing and the Price of Anarchy. MIT Press (2005)
5. Roughgarden, T.: On the severity of Braess's paradox: Designing networks for selfish users is hard. J. Comput. Syst. Sci. **72**(5) (2006) 922–953
6. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. Theoretical Computer Science **10**(2) (1980) 111–121
7. Schrijver, A.: Finding k disjoint paths in a directed planar graph. SIAM Journal on Computing **23**(4) (August 1994) 780–788
8. Barát, J.: Directed path-width and monotonicity in digraph searching. GRAPHS AND COMBINATORICS **22** (2006) 161–172
9. Robertson, N., Seymour, P.D.: Graph minors I. Excluding a forest. Journal of Combinatorial Theory **Series B**(35) (1983) 39–61
10. Bodlaender, H.L., Kloks, T.: Better algorithms for the pathwidth and treewidth of graphs. In: 18th International Colloquium on Automata, Languages and Programming. Volume 510 of LNCS., Springer Verlag (1991) 544–555