The Price of Optimum in Stackelberg Games on Arbitrary Single Commodity Networks and Latency Functions

A.C. Kaporis*
Department of Computer Engineering and Informatics, University of Patras
University Campus, Building B, GR 265 04, Patras, GREECE
kaporis@ceid.upatras.gr
Tel: +30 2610 996943, Fax: +30 2610 996980

P.G. Spirakis
Department of Computer Engineering and Informatics, University of Patras
& RA Computer Technology Institute,
N. Kazantzaki Str., Patra University Campus, 26500 Rio–Patra, GREECE
spirakis@cti.gr

Abstract

Let M be a single s-t network of parallel links with load dependent latency functions shared by an infinite number of selfish users. This may yield a *Nash* equilibrium with unbounded *Coordination Ratio* [23, 43]. A *Leader* can decrease the coordination ratio by assigning flow αr on M, and then all *Followers* assign selfishly the $(1-\alpha)r$ remaining flow. This is a *Stackelberg Scheduling Instance* $(M, r, \alpha),\ 0 \leq \alpha \leq 1$. It was shown [38] that it is weakly NP-hard to compute the optimal Leader's strategy.

For any such network M we efficiently compute the *minimum* portion β_M of flow $r > 0$ needed by a Leader to induce M's optimum cost, as well as her optimal strategy. This shows that the optimal Leader's strategy on instances $(M, r, \alpha \geq \beta_M)$ is in P.

Unfortunately, Stackelberg routing in more general nets can be arbitrarily hard. Roughgarden presented a modification of *Braess's Paradox* graph, such that *no* strategy controlling αr flow can induce $\leq \frac{1}{\alpha}$ times the optimum cost. However, we show that our main result also *applies* to any s-t net G. We take care of the Braess's graph explicitly, as a convincing example. Finally, we extend this result to k commodities.

A conference version of this paper has appeared in [16]. Some preliminary results have also appeared as technical report in [18].

Keywords:Stackelberg, Price of Optimum, Coordination Ratio

*Corresponding author.

1 Introduction

In large scale networks such as Internet the users/providers have freedom on how to route their load. This allows them to make their choices according to their own individual performance objectives, bringing the network to fixed points most times worse than the optimum one [11]. Such *selfish* behavior is being studied with the notion of *Nash Equilibrium* in the mathematical framework of Game Theory [9, 21, 22, 23, 27, 31, 32, 38, 45].

As a measure of how inefficient is the Nash equilibrium compared to the overall system's optimum, the notion of *coordination ratio* was introduced in the seminal paper of [23]. This work has been extended (*price of anarchy* is another equivalent term) in [7, 6, 14, 26, 34, 33, 42, 39, 38, 43, 36]. The interested reader can find nice presentations of this subject in the recent books [30, 42].

To improve the performance of the system under selfish behavior a great variety of methodologies have been considered so far. These methodologies intent to bring the system to fixed points closer to its optimum performance. The network administrator or designer can define prices, rules or even construct the network, in such a way that induces near optimal performance when the users selfishly use the system. This can be achieved through pricing policies [4], algorithmic mechanisms [12, 29, 28], network design [20, 36], or routing small portion of the traffic centrally [24, 21, 38].

Particulary interesting is the last approach where the network manager affects the non-cooperative game. The manager has the ability to control centrally a part of the system resources, while the remaining resources are used by the selfish users. This approach has been studied through *Stackelberg* or *Leader-Follower* games [24, 2, 21, 22, 38, 47]. One player (*Leader*) controls a portion of the system's jobs and assigns them to the system (*Stackelberg assignment*). The rest of the users (*Followers*) having in mind the assignment of the Leader react selfishly and reach a Nash equilibrium. The assignment of Leader and Followers is called *Stackelberg Equilibrium*. The goal of the Leader is to induce an optimal or near optimal Stackelberg Equilibrium.

1.1 Motivation & related work

(i) Single-commodity networks with parallel links. Consider a system M of parallel links and a total of flow $r > 0$ to be scheduled on M, denoted as a *Scheduling Instance* (M, r). Given an scheduling instance (M, r), there is a unique *Optimum* assignment O of flow $r > 0$ on system M minimizing the total cost $C(O)$ incurred on system M. We study the case of an infinite number of selfish users, each assigning its infinitesimal small portion of total flow $r > 0$ on links in M of currently minimum delay. Let the cost $C(N)$ of the *Nash* assignment N on the scheduling instance (M, r). Then,

$$C(N) = \epsilon_{(M,r)} \times C(O) \tag{1}$$

where $\epsilon_{(M,r)}$ depends only on instance (M, r) and can be arbitrarily larger than 1 [43], but if all links in M have linear load depended latency functions, then $\epsilon_{(M,r)} \leq 4/3$ [23]. We try to obtain a more clear picture of this degradation on system's performance, measured by the factor $\epsilon_{(M,r)}$, by studying *Stackelberg Scheduling Instances* as in [38], and as in [21] where we focus on the case of an infinite number of users. According to [38, 21] there is a central authority (*Leader*) that controls a portion $0 \leq \alpha \leq 1$ of the overall flow $r > 0$ to be assigned on system M, while the rest $(1-\alpha)r$ of the flow is assigned by the infinite self-optimizing users (*Followers*) on M. In [38] this is denoted as a *Stackelberg Scheduling Instance* $(M, r, \alpha), 0 \leq \alpha \leq 1$. This means that each scheduling instance (M, r) corresponds to a family of Stackelberg scheduling instances (M, r, α), parameterized with respect to $\alpha \in [0, 1]$. Given a Stackelberg scheduling instance (M, r, α), the

goal of the Leader is to find an assignment (*strategy*) S of his flow αr on M, such that to induce a Followers's assignment T of the remaining $(1-\alpha)r$ flow, with cost $C(S+T)$ near to the optimum $C(O)$ one. That is

$$C(S+T) \leq \epsilon_{(M,r,\alpha)} \times C(O) \tag{2}$$

Let us use the name "a-posteriori anarchy cost" for the quantity $\epsilon_{(M,r,\alpha)}$. Note that the a-posteriori anarchy cost depends on the strategy chosen by the Leader and on the portion α of the flow that she controls.

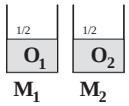


Figure 1: Latencies are: $\ell_1(x) = x, \ell_2(x) = 1$.

Figure 2: Leader's strategy is: $S = \langle 0, o_2 \rangle = \langle 0, 1/2 \rangle$.

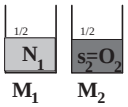
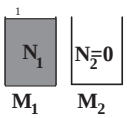


Figure 3: S induces NE $T = \langle o_1, 0 \rangle = \langle 1/2, 0 \rangle$.



> **Example [37, pp. 9]** (*Stackelberg parlance on Pigou's example*). In Fig. 1, link M_1 is faster than M_2, thus the selfish flow N floods M_1 (Fig. 1-down). A bad operating point arises: $C(N) > C(O)$, where the optimum O is to balance $r = 1$ to both links (Fig. 1-up). This induces the worst anarchy cost $\epsilon(M, 1) = 4/3$. A way out is in Fig. 2: *Leader* routes $1/2$ of r into slow link M_2. This is a wise strategy. Since, in Fig. 3, the remaining free $1/2$ flow induces the optimum operating point and *-the best possible-* a posteriori anarchy cost $\epsilon_{(M,1,1/2)} = 1$.

More precisely, in [42, Th. 6.4.4] it was proved that $\epsilon_{(M,r,\alpha)} \leq \frac{1}{\alpha}, 0 < \alpha \leq 1$ for arbitrary latency functions, and if restricted to instances with linear latencies then $\epsilon_{(M,r,\alpha)} \leq \frac{4}{3+\alpha}$ [42, Th. 6.4.5]. From Expression (2), we realize that the portion α captured by the Leader "pays" an upper bound on system's degradation factor $\epsilon_{(M,r,\alpha)}$ which is smaller than the plain one in Expression (1), see also [24]. More precisely, [38] presented the algorithm `LLF` that, on input a Stackelberg scheduling instance (M, r, α), computes a Leader's strategy S inducing Nash assignment T with performance guarantee $C(S+T) \leq \frac{1}{\alpha} C(O)$. However, on the same Stackelberg scheduling instance (M, r, α) there may exist a better Leader'strategy S' inducing T' such that $C(S'+T') < C(S+T)$, see footnote 6 in [38]. This means that `LLF` cannot always compute the optimal strategy. Also, there may exist a strategy S'', escaping from `LLF`, such that $C(S''+T'') = C(O)$. Such limitations are depicted in the negative result in [38] stating that the problem of computing the Optimal Stackelberg strategy on a given Stackelberg scheduling instance (M, r, α) is weakly NP-hard. A way out of this negative result was a FPAS devised in [24]. On approximating the optimal strategy, it is difficult to decide whether an arbitrary link receives a-posteriori induced flow or not, which reduces to a multidimensional knapsack problem in [24]. Our approach here eludes this difficulty by carefully identifying the corresponding subsets $M_{>0}(M_{=0})$ via recursively assigning "wise" amounts of flow per proper subset of links, see Section 7.3.

(ii) s-t **networks.** Finally, an important question of [38] that motivated us is the extension of the above results to arbitrary network graphs, closer to the nature of real networks. Given an arbitrary single source-destination (s, t)-network G, can a Leader wisely assign here α portion on some edges, inducing a selfish

$s \to t$ routing of the remaining flow with best possible cost? In [42, Example 6.5.1] it was exhibited a simple 4-nodes graph where no strategy can guarantee cost $\frac{1}{\alpha}$ times the optimum one. Notably, this 4-node graph is reminiscent to the one of *Braess's Paradox*. Before the publication of this work in [16, 18], no performance guarantee as a function of the centrally controlled portion α was known for (s,t)-nets.

After the publication of our work in [16, 18] a series of papers explored further important issues on Stackelberg routing on more general network topologies. For general networks and linear latencies, Karakostas and Kolliopoulos [19] bounded the anarchy cost via strategies SCALE and LLF. They also extended these results to more general latencies. In this vain, Swamy [46] obtained similar bounds for serial-parallel graphs with arbitrary latencies, while for more general graphs, he obtained latency-specific bounds on PoA. Recently, Fotakis [13] investigated similar strategies for atomic congestion games. The papers [5] and [3] are related to the aforementioned results, both investigating further the performance of strategy SCALE and a variation of LLF on general nets. Finally, our work is related to the recent paper of Sharma and Williamson [44]. They compute, on parallel links with linear latencies, the corresponding minimum portion of controlled flow sufficient to improve on the system's cost with respect to selfish routing.

(iii) Arbitrary multi-commodity nets. Even less was known for Stackelberg strategies on arbitrary networks before the publication of our work in [16, 18]. Some of the corresponding recent results in [3, 5, 19, 46] also extend to k commodities. More recently, [3] answered in the negative the [40, Open Problem 4] and improved the bounds in [46].

2 Problem definition & results

Problem: The input is a scheduling instance (M,r) where $r>0$ is the total flow and M is either m parallel links, or an s-t network, or, even a multi commodity net, with arbitrary load dependent edge latencies as in Sec. 4. The problem is to efficiently compute the *minimum* portion β_M of flow $r>0$ that a Leader must control, as well as her optimal strategy S, in order to induce the overall optimum cost $C(O)$ on (M,r). The main result is Theorem 2.1.

Theorem 2.1 *Let (M,r) be a multicommodity instance with k source/destination pairs s_i-t_i on an arbitrary network M, modeled as a directed graph. We can efficiently compute the minimum portion β_M of flow controlled by a Leader as to induce the optimum routing of total flow $r>0$ on instance (M,r). We also efficiently compute the associated optimal strategy of the Leader.*

The proof appears in Section 5.1.

Direct consequences of this theorem are Corollaries 2.2 (s-t parallel links) and 2.3 (s-t network). In particular, Corollary 2.2 shows that the algorithm `OpTop` efficiently solves the problem on parallel links. A helpful[1] illustration of `OpTop` appears in Section 3.1.

Algorithm: `OpTop` (M : parallel links, r : total flow.)
(1) Set $r_0 = r$ the total flow in M.
Compute the optimum assignment $O := \langle o_i : M_i \in M\rangle$ on instance (M, r_0). Set $M' \equiv \emptyset$.
(2) Compute the Nash assignment $N := \langle n_i : M_i \in M\rangle$ on instance (M,r).
(3) For each link $M_i \in M$ such that $o_i > n_i$ set $M' = M' \cup \{M_i\}$. If $M' \equiv \emptyset$ go to (5).
(4) Set $M = M \setminus M'$ and $O := O \setminus \{o_i \in M'\}$ and $r = r - \sum_{M_i \in M'} o_i$. Set $M' \equiv \emptyset$ and go to (2).
(5) The portion of flow controlled by the Leader is $\beta_M = (r_0 - r)/r_0$.

---

[1] At the end of this paper, Sec. 7 presents monotonicity properties of `OpTop`'s evolution. These properties may help to improve intuition on Stackelberg strategies.

Corollary 2.2 *Let (M, r) be an s-t parallel-links instance. Algorithm* `OpTop` *runs in polynomial time and computes the* minimum *portion β_M of total flow $r > 0$ that a Leader must control to induce overall optimum routing on (M, r), as well as the optimal Stackelberg strategy.*

Corollary 2.2 implies that on all Stackelberg scheduling instances of the form $(M, r, \alpha \geq \beta_M)$, a Leader can enforce the Optimum cost $C(O)$, and the problem of computing the Optimum Stackelberg strategy is in P. This eludes the hardness result in [41, Theorem 6.1] and finally answers an open question in [37, page 28]. In view of Expression (2), for such instances the factor $\epsilon_{(M,r,\alpha\geq\beta_M)}$ is precisely 1. On the contrary, for all Stackelberg sceduling instances $(M, r, \alpha < \beta_M)$ it is not possible for a Leader to enforce the Optimum cost. Then in Expression (2) we get that $\epsilon_{(M,r,\alpha<\beta_M)} > 1$, which means that such Stackelberg scheduling instances are the really hard ones and we can try to attack these by sophisticated fully polynomial approximation schemes [24]. Such non-optimizing behavior was presented also in [21], for the restricted case of M/M/1 systems of *distinct* links. Notably, if such M/M/1 systems contain small groups of highly appealing links or there are large groups of identical links then β_M may be significantly small.

Generalizing the previous algorithm on s-t nets, we get algorithm `MOP`, where in turn, Corollary 2.2 shows that it efficiently solves the problem. A helpful illustration of `MOP` appears in Section 3.2.

Algorithm: `MOP` (G : an s-t net, r : total flow)
(1) Initialize Stackelberg strategy $S = \{\}$, centrally captured flow $r_S = 0$.
(2) Compute the optimum assignment $O := \langle o_e : e \in G \rangle$ on instance (G, r).
(3) Set cost $\ell_e(o_e)$ on each edge $e \in G,\ o_e \in O$.
(4) Compute the shortest paths in G with edge-costs $\ell_e(o_e),\ e \in G$. Let $\mathcal{P}^O_{s\to t}$ the set of shortest paths.
(5) Control flow $O_P > 0$ on each non shortest path $P \notin \mathcal{P}^O_{s\to t}$.
(6) Set r' the uncontrolled flow which routes through shortest paths $\mathcal{P}^O_{s\to t}$.
(7) Return the Leader's portion $\beta_G = 1 - \frac{r'}{r}$.

Corollary 2.3 *Let instance (M, r) on a s-t network M. Algorithm* `MOP` *computes in polynomial time the minimum portion β_M of Leader's flow, sufficient to induce the optimum routing of flow $r > 0$ on network G, as well as the optimal strategy of the Leader.*

Corollary 2.3 implies that, despite the negative result depicted in [42, Example 6.5.1], algorithm `MOP` efficiently computes the optimal strategy on arbitrary nets, yielding approximation guarantee 1, see in this respect the open problem in [37, page 29]. In particular, we take care this bad example of net routing in Section 3.2.

On hard instances of parallel-links. Trying to understand further the underlying complexity of hard instances $(M, r, \alpha < \beta_M)$, we focus on parallel links, with *appropriate* load dependent latency functions that, hopefully, may admit efficient computation of the optimal strategy. Our motivation is the case of simple followers (which is identical to an infinite number of followers that we consider here) studied in Section 8 in [21].

Theorem 2.4 *The optimal Stackelberg strategy is computed in polynomial time on any instance $(M, r, \alpha < \beta_M)$ with m parallel links where each link has linear latency $\ell_i(x) = ax + b_i, a \geq 0, b_i \geq 0, i = 1, \ldots, m$.*

The proof appears in Section 6.1. This theorem squeezes "efficiency" from the class of linear latencies, shown hard in the reduction in [41, Theorem 6.1].

Remark 2.5 *The results of this paper concern instances with strictly increasing latencies, pervasive in real-world networks, see the classical [1] work of Patriksson [35, Ch. 4] and also the pioneering work of Dafermos et al. [8, Sec. 1.3]. A property important to our analysis, is that on such instances Optimum O and NE N edge flows are unique. In [17] we extend such instances, while preserving uniqueness of optimum edge flows, by even allowing edges with constant latencies.*

3 Examples on computing the price of Optimum

3.1 s-t parallel links

Algorithm `OpTop` of Corollary 2.2 works as follows:

1. Compute the Optimum $O := \langle o_1, \ldots, o_m \rangle$ (Fig. 4-up) & Nash $N := \langle n_1, \ldots, n_m \rangle$ (Fig. 4-down).
2. Detect the under-loaded (Def. 4.3) links $M_i \in M$ with $n_i < o_i$ (Fig. 4, links M_4, M_5).
3. Play Stackelberg strategy $s_i = o_i$ per under-loaded link $M_i \in M$ (Fig. 5-up).
4. Discard these under-loaded links and the flow assigned to them (Fig. 5-down).
5. Assign the remaining flow recursively to the simplified subnetwork of links.
6. Terminate if the simplified subnetwork has all the links optimum-loaded (Fig. 6).

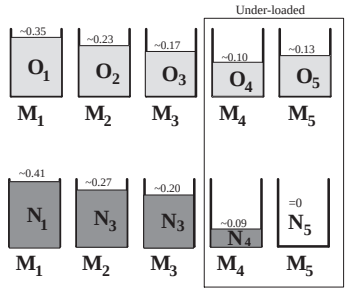


Figure 4: $\ell_1(x) = x, \ell_2(x) = 3/2x, \ell_3(x) = 2x, \ell_4(x) = 5/2x + 1/6, \ell_5(x) = 7/10$.

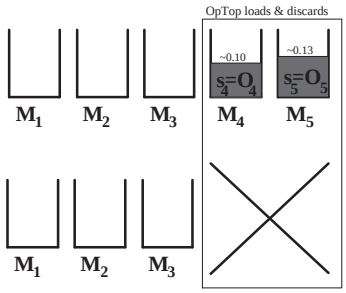


Figure 5: **Up:** `OpTop` optimally loads M_4, M_5. **Down:** `OpTop` discards M_4, M_5.

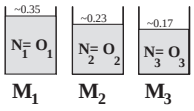


Figure 6: `OpTop` terminates: the remaining $1 - O_4 - O_5$ selfish flow just induced NE equal to the Optimum.

3.2 s-t networks

Let $\mathcal{P}_{s \to t}$ be the $s \to t$ paths on instance (G, r). Algorithm `MOP` of Corollary 2.3 works as follows:

1. Compute the optimum flow O, assigning flow o_e and latency $\ell_e(o_e)$ per edge $e \in G$ (Fig. 7-(a)).
2. Compute the shortest paths $\mathcal{O}_{s \to t} \subseteq \mathcal{P}_{s \to t}$ with respect to costs $\ell_e(o_e), \forall e \in G$, (see Fig. 7-(b)).
3. Play the Stackelberg strategy assigning the optimal flow $O_P > 0$ on each path $P \notin \mathcal{O}_{s \to t}$ (Fig. 7-(c)).
4. The Price of Optimum $\beta_G =$ (the optimal flow on all non-shortest paths)$/r$ (Fig. 7-(d)).

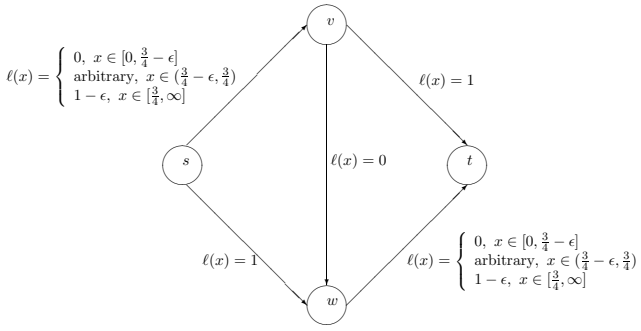


Figure 7: The total flow is $r = 1$. Details on this instance can be found in [42, Example 6.5.1].
(a) The optimal edge-flows are: $o_{s\to v} = \frac{3}{4} - \epsilon, o_{s\to w} = \frac{1}{4} + \epsilon, o_{v\to w} = \frac{1}{2} - 2\epsilon, o_{v\to t} = \frac{1}{4} + \epsilon, o_{w\to t} = \frac{3}{4} - \epsilon$.
(b) The shortest path induced by optimum flow O is: $P_0 = s \to v \to w \to t$, with flow $O_{P_0} = \frac{1}{2} - 2\epsilon$.
(c) Non-shortest paths are: $P_1 = s \to v \to t$ & $P_2 = s \to w \to t$, with flows $O_{P_1} = \frac{1}{4} + \epsilon$ and $O_{P_2} = \frac{1}{4} + \epsilon$.
(d) The Price of Optimum is: $\beta_G = \frac{r - O_{P_0}}{r}$.

Remark 3.1 *In s-t parallel links with arbitrary latency functions, a simple algorithm controls $\alpha \in [0, 1]$ portion of flow and induces an equilibrium with cost $\leq \frac{1}{\alpha}$ times the optimum one ($\frac{1}{\alpha}$ approximation guarantee). Roughgarden showed [42, Example 6.5.1] that the above bound is not possible to hold for s-t networks. This was demonstrated with a selfish routing example on the graph in Fig. 7. The importance of our example here stems from the fact that on this particular graph in Fig. 7,* `MOP` *achieves the optimum cost (thus, its approximation guarantee equals 1, which is $\leq \frac{1}{\alpha}, \forall \alpha \in [0, 1)$).*

4 Model

We briefly present the model below, discussed thoroughly in [42].

s-t parallel links: We have m parallel links $M = \{M_1, \ldots, M_m\}$ connecting a source s to a sink vertex t. An infinite number of users wish to route from s to t, each controlling an infinitesimal small portion of a total flow $r > 0$. Each link $M_i \in M$ on flow x_i has *standard* latency function: $\ell_i(x_i) \geq 0$ differentiable, strictly increasing[2] and $x_i \ell_i(x_i)$ convex on x_i. Any assignment of jobs to the links in M is represented as a feasible m-vector $X = \langle x_1, \ldots, x_m \rangle \in \mathcal{R}^m$ such that $X \geq 0$ and $\sum_{i=1}^m x_i = r$. A routing instance is annotated as (M, r). The *Cost* of a feasible assignment $X \in \mathcal{R}^m$ on instance (M, r) equals $C(X) = \sum_{i=1}^m x_i \ell_i(x_i)$. The minimum cost is incurred by a unique feasible assignment $O \in \mathcal{R}^m$ called the *Optimum*. The unique feasible assignment $N \in \mathcal{R}^m$ defines a *Nash Equilibrium* (NE), if no user can find a link with latency $<$ than any other loaded link. A consequence is:

Remark 4.1 *All loaded links by N have latency L^N, while the empty ones have latency $\geq L^N$, where $L^N > 0$ is an appropriate constant.*

A *Stackelberg* strategy $S = \langle s_1, \ldots, s_m \rangle$ is a feasible flow of a portion βr of $r > 0$ on $M, \beta \in [0, 1]$. In other words, $\sum_{i=1}^m s_i = \beta r$ and $s_i \geq 0$ per link $M_i \in M$. Let $T = \langle t_1, \ldots, t_m \rangle$ be the *induced NE* (INE) by strategy S. Hence, T satisfies $\sum_{i=1}^m t_i = (1 - \beta) r$ with $t_i \geq 0$ per link $i \in M$, and by rephrasing Remark 4.1, we get:

[2] See Remark 2.5.

Remark 4.2 *All loaded links by T have latency L^S, while the empty ones (by T) have latency $\geq L^S$, where $L^S > 0$ is an appropriate constant depending on S.*

The flow $S + T$ is called *Stackelberg Equilibrium* with cost $C(S+T) = \sum_{i=1}^{m}(s_i + t_i)\ell_i(s_i + t_i)$.

Definition 4.3 *Link $M_i \in M$ is called* over-loaded *if $n_i > o_i$,* under-loaded *if $n_i < o_i$, otherwise is called* optimum-loaded*, $i = 1, \ldots, m$.*

Definition 4.4 *Link $M_i \in M$ is* frozen *if Stackelberg strategy S assigns to it load $s_i \geq n_i$ where N is the initial Nash assignment. Otherwise M_i is* unfrozen*.*

Multicommodity networks: A network can be modeled as a directed graph $G(V, E)$ with set of vertices V and edges E. There are k source-destination pairs of vertices[3] $(s_1, t_1), \ldots, (s_k, t_k)$ and no self loops are allowed. $\mathcal{P}_i$ is the set of all paths amongst $(s_i, t_i), i = 1, \ldots, k$, and $\mathcal{P} = \bigcup_i \mathcal{P}_i$. An infinite number of users wish to route from each s_i to t_i, each controlling an infinitesimal small portion of total flow $r_i > 0$. Let $f = \langle f_P : P \in \mathcal{P} \rangle$ be a flow $r = \sum_{i=1}^{k} r_i$ to the paths in $\mathcal{P}$, where f_P denotes the flow traveling through path $P \in \mathcal{P}$. The flow f_e on edge $e \in E$ is the flow it receives from all paths in $\mathcal{P}$ traversing e. On a pair (s_i, t_i), we let f^i be the restriction of f to $\mathcal{P}_i, i = 1, \ldots, k$. The total of flow wishing to travel through source-destination pair (s_i, t_i) is r_i and f is feasible if the flow it assigns on the paths in $\mathcal{P}_i$ is r_i. Edges are endowed with latency functions as in the parallel-links model above[4]. The latency of a path $P \in \mathcal{P}_i$ with respect to flow f is the sum of its edge-latencies $\ell_P(f) = \sum_{e \in P} \ell_e(f_e)$. The cost of a flow f is $C(f) = \sum_{e \in E} \ell_e(f_e) f_e = \sum_{P \in \mathcal{P}} \ell_P(f) f_P$. The unique *Optimal* flow O is the one minimizing the cost $C(\cdot)$ of scheduling flow $r > 0$ on network G and can be efficiently computed. A feasible flow N is a Nash equilibrium on G if an only if for every commodity $i \in \{1, \ldots, k\}$ and paths $P_1, P_2 \in \mathcal{P}_i$ with $f_{P_1} > 0$ we have $\ell_{P_1}(f) \leq \ell_{P_2}(f)$. In other words, in each source-destination pair (s_i, t_i) no flow traveling through loaded path $P_1 \in \mathcal{P}_i$ can find any other path $P_2 \in \mathcal{P}_i$ with $<$ latency. A *Stackelberg* strategy $S = \langle s_P : P \in \mathcal{P} \rangle$ on instance (G, r) is a feasible assignment of a portion $\beta r, \beta \in [0, 1]$ of the total flow $r > 0$ on the paths in $\mathcal{P}$. A Leader dictates a *weak* Stackelberg strategy if on each commodity $i = 1, \ldots, k$ controls a fixed α portion of flow r_i, $\alpha \in [0, 1]$. Also, on a *strong* Stackelberg strategy a Leader controls $\alpha_i r_i$ flow in commodity i such that $\sum_{i=1}^{k} \alpha_i = \alpha$. Let a Leader dictating flow s_e on edge $e \in E$. Let $\mathcal{T} = \langle \tau_1, \ldots, \tau_m \rangle$ be the *induced NE* (INE) by strategy S of the remaining $(1 - \beta)r$ selfish flow. The a posteriori latency $\widetilde{\ell}_e(\tau_e)$ of edge e, with respect to the induced selfish flow $\mathcal{T}$, equals $\widetilde{\ell}_e(\tau_e) = \ell_e(\tau_e + s_e)$. In the a-posteriori Nash equilibrium $\mathcal{T}$, all s_i-t_i paths traversed by the selfish users in commodity i have a common latency, which is at least the latency of any empty path in commodity $i = 1, \ldots, k$. The induced cost of flow $S + \mathcal{T}$ is $\sum_{e \in E}(\tau_e + s_e) \times \widetilde{\ell}_e(\tau_e)$.

Remark 4.5 *In [13, 24, 41] the NE N and Optimum O flows are given as input. On the basis these flows, subsequent Stackelberg strategies are efficiently computed. In general, O and N flows can be efficiently computed and more details can be found in [42, Fact 2.4.9, Cor. 2.6.7], see also [42, Sec. 2.8]. For strictly increasing latencies, the corresponding edge-flows are unique [42, Cor. 2.6.4]. This extends to real-world instances endowed with edges with increasing latencies [17].*

5 Proof of the main result on k commodities

In this section we prove main Theorem 2.1, which yields Cor. 2.2 and 2.3.

[3] s_i denotes a source vertex, while s_e the Leader's flow on edge e and their meaning will be clear from the context.

[4] See Remark 2.5.

5.1 Proof of Theorem 2.1

Let $\mathcal{P}^{(i)}_{s\to t}$ be the set of all s_i-t_i paths on a k-commodity instance G, $i = 1, \ldots, k$. On commodity i, we efficiently compute[5] the subset $\mathcal{P}^{O,(i)}_{s\to t}$ of shortest paths $\mathcal{P}^{O,(i)}_{s\to t} \subseteq \mathcal{P}^{(i)}_{s\to t}$ with respect to optimum costs $\ell_e(o_e), \forall e \in G$.

Leader must control under strategy S the optimum flow O_P assigned by O on every *non* shortest path $P \in \mathcal{P}^{(i)}_{s\to t} \setminus \mathcal{P}^{O,(i)}_{s\to t}, \forall i$.

First, suppose that exists path $P \in \mathcal{P}^{(i)}_{s\to t}$ such that Leader controls flow $S_P > O_P$ under strategy S. Then the induced flow $S + T$ will *differ* from the unique optimum flow O at least on this particular path P, yielding suboptimal cost. Therefore it is meaningful for Leader, $\forall P \in \mathcal{P}^{(i)}_{s\to t}$, to assign only flow $S_P \leq O_P$ under any strategy S she employs.

Now, suppose that in commodity i there exists non shortest path $P_0 \in \mathcal{P}^{(i)}_{s\to t} \setminus \mathcal{P}^{O,(i)}_{s\to t}$ such that Leader controls flow $S_{P_0} < O_{P_0}$ under strategy S. The remaining uncontrolled flow $O_{P_0} - S_{P_0}$ on this non shortest path will opt for a currently shortest path in $\mathcal{P}^{O,(i)}_{s\to t}$. Then the induced flow $S + T$ will *differ* from the unique optimum flow O on this sub-optimally loaded path P_0. Furthermore, Leader need not waste any flow $S_P \leq O_P$ on any shortest path $P \in \mathcal{P}^{O,(i)}_{s\to t}$, because in this way path P still remains a shortest one, and thus, any $O_{P_0} - S_{P_0}$ uncontrolled flow in the aforementioned non shortest P_0 will opt for P, yielding $S+T \neq O$.

It follows that the only choice for the Leader is to optimally load every non shortest path $P \in \mathcal{P}^{(i)}_{s\to t} \setminus \mathcal{P}^{O,(i)}_{s\to t}$ per commodity i.

6 s-t parallel links on hard instances $(M, r, \alpha < \beta_M)$

Here we prove Theorem 2.4. Consider an instance (M, r) of m parallel links, with latency functions $\ell_i(x) = ax + b_i, a \geq 0, b_i \geq 0, i = 1, \ldots, m$. With no loss of generality, assume that the constant terms of the latencies are strictly increasing, that is, $b_i < b_{i+1}$, per link $i = 1, \ldots, m-1$.

In Lemma 6.1 we show that on any instance $(M, r, \alpha < \beta_M)$ there is a Leader's optimal strategy that partitions $M = \{M_1, \ldots, M_{i_0}, \ldots, M_m\}$ around some link M_{i_0} such that subsystem $M_{>0}(i_0)$ (containing links appealing to Followers) is $M_{>0}(i_0) = \{M_1, \ldots, M_{i_0}\}$ and subsystem $M_{=0}(i_0)$ (containing links that Followers dislike) is $M_{=0}(i_0) = \{M_{i_0+1}, \ldots, M_m\}$.

Lemma 6.1 *There exists an optimal strategy of S of the leader, such that all links in $M_{=0}$ have indices greater than ones of the links in $M_{>0}$. This strategy can be computed in polynomial time.*

Proof. Let S be an optimal strategy for the Leader and T the corresponding induced Nash assignment. Suppose that S does not satisfy the property of Lemma 6.1, with no loss of generality, assume that $M_1 \in M_{=0}$ (which means $t_1 = 0$) and $M_2 \in M_{>0}$ (which means $t_2 > 0$), see Fig. 8. The partial cost of $S + T$ on subsystem $\{M_1, M_2\}$ equals:

$$\begin{aligned} s_1\ell_1(s_1) + (s_2 + t_2)\ell_2(s_2 + t_2) &= \\ s_1\ell^1 + (s_2 + t_2)\ell^2 &= A \\ \text{with } \ell^1 \geq \ell^2 & \end{aligned} \tag{3}$$

[5]Implementing *Dijkstra*'s algorithm, for example as in [10], compute subgraph $\mathcal{G} \subseteq G$ containing all edges traversed by a shortest path with respect to edge costs incurred by O. Per source s_i of commodity i, compute (free flow) the flow by O going through edges in $\mathcal{G}$ adjacent to s_i. This gives the controlled flow $\beta_G r$, which is r minus the total free flow. Leader assigns this $\beta_G r$ controlled flow on G by loading $s_e = o_e, \forall e \notin \mathcal{G}$, satisfying the standard multicommodity flow constraints.

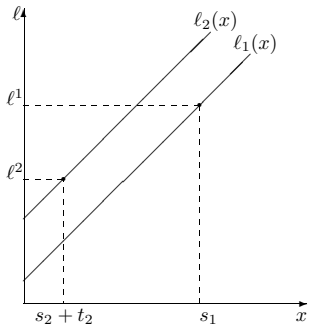


Figure 8: The loads and latencies on M_1 and M_2 links: load s_1 encounters latency ℓ^1 on link M_1 & load $s_2 + t_2$ encounters latency $\ell^2 \leq \ell^1$ on link M_2.

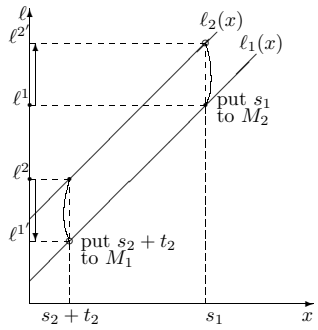


Figure 9: Interchange M_1's and M_2's loads: s_1 goes to M_2 increasing the latency to $\ell^{2\prime} > \ell^1$ while $s_2 + t_2$ goes to M_1 decreasing the latency to $\ell^{1\prime} < \ell^2$.

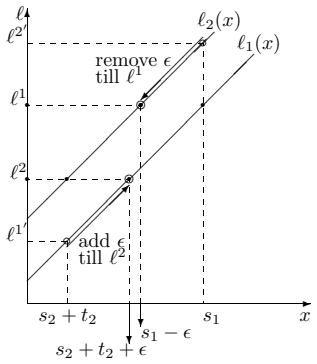


Figure 10: Move load ϵ from M_2 (decreasing M_2's latency to M_1's old one ℓ^1) to M_1 (increasing M_1's latency to M_2's old one ℓ^2). The cost is $\leq$ in Fig. 8

We show how to reassign the Leader's flow $s_1 + s_2$ on subsystem $\{M_1, M_2\}$, in a way that $t_1 > 0$ (hence satisfying the property of Lemma 6.1), inducing partial cost $\leq A$ and latency $\leq \ell^2$ on the most appealing link. In this way, we get a new Leader strategy, with cost $\leq A$, that *satisfies* the property described in Lemma 6.1.

First, interchange the loads between M_1 and M_2, i.e. load s_1 goes from M_1 to M_2 and load $s_2 + t_2$ goes from M_2 to M_1, see Fig. 9. This decreases the latency on M_1 to $\ell^{1'} < \ell^2$. Also it increases the latency on M_2 to $\ell^{2'} > \ell^1$. Since the latency functions have parallel plots, we can remove load $\epsilon \geq 0$ from M_2 till the latency it experiences drops from $\ell^{2'}$ to ℓ^1 and place it to M_2 raising its latency from $\ell^{1'}$ to ℓ^2, see Fig. 10. The new cost on $\{M_1, M_2\}$ is:

$$\begin{aligned}(s_2 + t_2 + \epsilon)\ell_1(s_2 + t_2 + \epsilon) + (s_1 - \epsilon)\ell_2(s_1 - \epsilon) &= \\ (s_2 + \epsilon + t_2)\ell^2 + (s_1 - \epsilon)\ell^1 = A + \epsilon(\ell^2 - \ell^1) &\leq A, \\ \text{since } \ell^2 \leq \ell^1, \epsilon \geq 0 &\end{aligned}$$

■

6.1 Proof of Theorem 2.4

Given an instance (M, r, α), fix a partition $M_{>0}(i_0) = \{M_1, \ldots, M_{i_0}\}$ and $M_{=0}(i_0) = \{M_{i_0+1}, \ldots, M_m\}$ of system M (by Lemma 6.1 there are $\leq m$ such partitions). We wish to find $0 \leq \epsilon_{i_0} \leq \alpha r$ such that the partial cost on subsystem $(M_{>0}(i_0), (1-\alpha)r + \epsilon_{i_0})$ by *selfishly* routing $(1-\alpha)r + \epsilon_{i_0}$ flow on it (such that no machine in $M_{>0}(i_0)$ remains empty), added to the partial cost on subsystem $(M_{=0}(i_0), \alpha r - \epsilon_{i_0})$, when *optimally* assigning $\alpha r - \epsilon_{i_0}$ flow on it, is minimized. The constraint is that the common latency in $(M_{>0}(i_0), (1-\alpha)r + \epsilon_{i_0})$ must be at most the latency of any link in $(M_{=0}(i_0), \alpha r - \epsilon_{i_0})$, otherwise users in $M_{>0}(i_0)$ will opt for machines in $M_{=0}(i_0)$ destroying the induced assignment.

The cost on subsystem $(M_{>0}(i_0), (1-\alpha)r + \epsilon_{i_0})$ reduces to computing the cost of the *Nash* assignment of flow $(1-\alpha)r + \epsilon_{i_0}$ onto subsystem $M_{>0}(i_0)$. The cost on subsystem $(M_{=0}(i_0), \alpha r - \epsilon_{i_0})$ reduces to computing the *Optimum* assignment of flow $\alpha r - \epsilon_{i_0}$ onto subsystem $M_{=0}(i_0)$. Such computations can be done efficiently for linear latencies. The addition of both costs is acceptable, only if for the given value of ϵ_{i_0}, *all* links in $M_{>0}(i_0)$ become loaded, and their common latency $L^{\epsilon_{i_0}}$ is at most the minimum latency experienced in any link in $M_{=0}(i_0)$ (in case of violation of any such constraint, we set the cost equal to ∞).

Notice here that for a given $(M, r, \alpha < \beta_M)$ and any index $i_0 \in \{1, \ldots, m-1\}$, the partial cost on $(M_{>0}(i_0), (1-\alpha)r + \epsilon_{i_0})$ is strictly increasing on ϵ_{i_0}, while the partial cost on $(M_{=0}(i_0), \alpha r - \epsilon_{i_0})$ is strictly decreasing on ϵ_{i_0}. This allows as to efficiently compute the value $\epsilon^*_{i_0}$ that minimizes the sum $C(\epsilon^*_{i_0})$ of the partial costs.

The optimal Leader strategy is determined by the tuple $(\iota, \varepsilon_\iota), \iota \in \{1, \ldots, m-1\}, 0 \leq \varepsilon_\iota \leq \alpha r$ such that $C(\varepsilon_\iota) = \min\{C(\epsilon^*_1), \ldots, C(\epsilon^*_{m-1})\}$. ∎

7 Miscellaneous: more on `OpTop` & parallel links

The hardness of computing the optimal Leader's strategy stems from a variant of the PARTITION problem ([41, Theorem 6.1]). Intuitively, the exponentially many ways for the Leader to place her αr flow on the m links make elusive the identification of the optimal strategy.

Section 7.2 helps to significantly prune the search space, where Theorem 7.2 identifies all useful strategies. Any useful strategy S induces Nash T with cost $C(S+T) \neq C(N)$, N is the initial equilibrium when all users are free. According to this theorem[6], $C(S+T) \neq C(N)$ holds only if strategy S assigns load $s_j > n_j$ to at least one link M_j (freezes link M_j, Def. 4.4), where n_j equals the initial Nash load.

Theorem 7.2 forces Leader to freeze at least one link (assign more than Nash load), otherwise the induced cost $C(S+T)$ equals the Nash cost $C(N)$.

Having played S, Leader now faces the difficulty of identifying, amongst the links loaded by S, those that will (or not) receive induced selfish flow (see the discussion in [24, Section 3.2, 2nd paragraph] and also here in Section 7.3) and subsequently ruin (or maintain) her assignment. The way out is presented in Section 7.3. In this section, Theorem 7.4 and Lemma 7.5 will help Leader escape this difficulty and will allow her to dynamically discover her optimal strategy as `OpTop` evolutes. Theorem 7.4 and its extension Lemma 7.5 prove that as soon as a link M_j gets load $s_j \geq n_j$ (link M_j becomes frozen) then no induced selfish load ($t_j = 0$) will be assigned to it, irrespectively of the assignment S to the remaining links.

Leader takes advantage of the discussions in Sections 7.2 and 7.3 and exploits link-by-link her optimal strategy S, as described in Section 7.4: Leader is forced by Theorem 7.2 to overload $s_j > n_j$ at least one link M_j (otherwise the induced cost remains $C(N)$). Any frozen link M_j must be optimum-loaded $s_j : n_j < s_j = o_j$ (Definition 4.3). Otherwise, the flow $s_j : n_j < s_j \neq o_j$ it receives is not possible to be affected by the induced selfish play, as Theorem 7.4 and Lemma 7.5 demonstrate, implying suboptimal $C(S+T) > C(O)$ induced cost (since $\exists j : s_j \neq o_j$).

Hence, the only option for the Leader is to recursively assign optimal load $s_j = o_j$ to any link M_j currently under-loaded $n_j < o_j$ and simplify the game by removing such M_j's, see Section 7.4, which combines the results of Sections 7.2 and 7.3.

[6] After the publication of our work a more clear picture on the property $C(S+T) \neq C(N)$ was provided. In [44, Eq. (1)] it was shown that any useful strategy S inducing $C(S+T) < C(N)$ *must* control at least the *minimum* Nash load $\min_{i=1,\ldots,m}\{n_i | n_i < o_i\}$ on *any* under-loaded link M_i (Def. 4.3), $i = 1, \ldots, m$.

7.1 A monotonicity property

Now we give a simple proof for a monotonicity property that will be important to our proof methodology. A similar property was proved in [25, Theorem 3] via alternating paths for general s-t nets. Previously, this was also shown with methods of sensitivity analysis in [15]. After the publication of our work, the authors in [44] gave an alternative proof.

Proposition 7.1 *If on $(M, r'), (M, r)$ holds $r' \leq r$, for the respective NEs N, N' holds $n'_i \leq n_i, \forall M_i \in M$.*

Proof. N is Nash equilibrium of flow r on M and by Definition 4.1, $\exists\ L^N > 0$, such that for each link $M_i \in M$ if $n_i > 0$ then $\ell_i(n_i) = L^N$, otherwise $\ell_i(n_i) = \ell_i(0) \geq L^N$. Let

$$M^{N^+} = \{M_i \in M : n_i > 0\} \text{ and } M^{N^-} = \{M_i \in M : n_i = 0\} \tag{4}$$

Similarly for equilibrium N' of the flow r', let $L^{N'} > 0$ the corresponding constant. To reach a contradiction, suppose that $\exists\ M_{i_0} \in M$ such that $n'_{i_0} > n_{i_0}$. Then $\ell_{i_0}(n'_{i_0}) = L^{N'} > \ell_{i_0}(n_{i_0}) \geq L^N$, since each $\ell_i(\cdot)$ is strictly increasing. Then, each link $M_i \in M^{N^+}$ must have load $n'_i > n_i$ under N', otherwise it will experience latency $\ell_i(n'_i) \leq \ell_i(n_i) = L^N < L^{N'}$ which is impossible, since N' is a Nash equilibrium. Therefore, we reach a contradiction since we get $r' \geq \sum_{M_i \in M^{N^+}} n'_i > \sum_{M_i \in M^{N^+}} n_i = r$. ∎

7.2 Identifying the useful strategies

Luckily, by the Nash assignment N of the users, all links may end up optimum-loaded (Def. 4.3). In this way, $N \equiv O$ and the cost $C(N)$ of the system is minimized, that is $C(N) = C(O)$. In general $N \not\equiv O$, since the selfish users prefer and thus over-load fast links, while dislike and under-load slower ones, increasing the cost $C(N) > C(O)$. The crucial role of strategy S is to wisely pre-assign load $s_i \geq 0$ to each link $M_i \in M$. This is successful to the extent that the induced Nash assignment T made by the users will assign an additional load $t_i \geq 0$ to each M_i, yielding the nice property $s_i + t_i = o_i$ for each $i = 1, \ldots, m$. Intuitively, strategy S *biasses* the initial Nash assignment N to the induced one T, in a way that $S + T \equiv O$, minimizing the induced overall cost $C(S + T) = C(O)$ on system M.

Theorem 7.2 describes each Stackelberg strategy S inducing Nash assignment T with cost $C(S + T) = C(N)$. In other words, Theorem 7.2 describes exactly all those useless strategies that induce cost indifferent from $C(N)$. Then, it is useless for `OpTop` to employ such a strategy S when trying to escape from a particular Nash equilibrium N with $C(N) > C(O)$.

Theorem 7.2 *If for strategy S holds $\forall j, s_j \leq n_j$ then for the induced Nash assignment T it holds $\forall j, n_j = s_j + t_j$. In other words, the initial Nash assignment N will coincide to $S + T$.*

Proof. Since N is a Nash equilibrium on the links in M with $\sum_{i=1}^{m} n_i = r$, then there exists a constant $L^N > 0$, such that for each link $M_j \in M$ that receives load $n_j > 0$ it holds $\ell_j(n_j) = L^N$. That is, all loaded links incur the same latency L^N to the system M. Consider an arbitrary Stackelberg strategy S, assigning load $s_j \leq n_j$ to each $M_j \in M$ with $\sum_{i=1}^{m} s_i = \beta r, \beta \in [0,\ 1]$. Since for each $M_j \in M$ it holds $s_j \leq n_j$ then $\exists\ t_j \geq 0$ such that $t_j = n_j - s_j$ and also $\sum_{i=1}^{m} t_i = (1 - \beta)r$. Let $T = \langle t_1, \ldots, t_m \rangle$ be this assignment on M. Obviously, for the same constant $L^N > 0$ as above, it holds: $\ell_j\left((n_j - s_j) + s_j\right) = \ell_j(n_j) = L^N$, for each $M_j \in M$ with $t_j > 0$. This means that T is a Nash equilibrium on system M and also $S + T \equiv N$. ∎

Definition 7.3 *Each Stackelberg strategy S that satisfies Theorem 7.2 is called* useless-strategy*, otherwise is called* useful-strategy.

7.3 Each frozen link gets no induced selfish flow

It is helpful to quote here the main difficulty on approximating the optimal strategy, as described in [24, Sections 4.1-4.2]. Let S^* an optimal strategy, T^* the induced Nash, and subset $M_{>0}(M_{=0})$ of links that do (not) get induced selfish flow. The hard part is to decide whether link i belongs to $M_{>0}$ or to $M_{=0}$. By observing that all links in $M_{>0}$ (or $M_{=0}$) have common latency $\ell_i(s_i + t_i) = L^*$ (or common marginal cost $(s_i\ell_i(s_i))' = D^*$), the authors in [24] reduce such decisions to a multidimensional knapsack problem.

This section eludes this difficulty by carefully identifying the corresponding subsets $M_{>0}(M_{=0})$ via recursively assigning "wise" amounts of flow per proper subset of links. Intuitively, subset $(M_{>0})M_{=0}$ stands for (un)frozen links, defined bellow.

In the sequel, Theorem 7.4 and Lemma 7.5 demonstrate an invariant property of frozen links: a frozen link belongs to $M_{=0}$, irrespectively of Leader's play to remaining links.

Theorem 7.4 *If strategy S freezes (Def. 4.4) every link it assigns flow then all frozen links get no induced selfish flow.*

Proof. Let the Nash assignment $N = \langle n_1, \ldots, n_m\rangle$ on instance (M, r). Strategy $S = \langle s_1, \ldots, s_m\rangle$ freezes every link it assigns load, that is, either $s_j \geq n_j$ or $s_j = 0, \forall j$. We show that on each frozen link $M_j \in M$ the induced Nash assignment T assigns flow $t_j = 0$.

N is a Nash equilibrium, and by Definition 4.1, $\exists\ L^N > 0 : \forall M_i \in M$, if $n_i > 0$ then $\ell_i(n_i) = L^N$, otherwise $\ell_i(n_i) \geq L^N$. Fix a Stackelberg strategy S and let the subsystems of frozen and unfrozen links:

$$M^{S^+} = \{M_i \in M : s_i \geq n_i\} \text{ and } M^{S^-} = \{M_i \in M : s_i = 0\} \tag{5}$$

partitioning system M. Each frozen link $M_i \in M^{S^+}$ receiving induced load $t_i \geq 0$ experiences a posteriori higher latency (less appealing) than the initial common value L^N, since:

$$\ell_i^{S^+}(t_i) = \ell_i(t_i + s_i) \geq \ell_i(s_i) \geq \ell_i(n_i) \geq L^N \tag{6}$$

On the contrary, on each unfrozen link $M_j \in M^{S^-}$ its a posteriori latency function remains the *same* as before applying strategy S (since $s_j = 0$):

$$\ell_j^{S^-}(t_j) = \ell_j(t_j + s_j) = \ell_j(t_j) \tag{7}$$

The induced Nash assignment T by strategy S assigns the remaining selfish flow on M. This flow is $\leq$ the initially $\sum_{M_j \in M^{S^-}} n_j$ selfish flow assigned under N only on unfrozen links in M^{S^-}, since by the definition of M^{S^+}, M^{S^-} in (5) it holds:

$$r - \sum_{i=1}^{m} s_i = r - \sum_{M_j \in M^{S^+}} s_j \leq \sum_{M_j \in M^{S^-}} n_j \tag{8}$$

Taking advantage (7) and (8), Proposition 7.1 implies that the remaining free flow that appears in LHS of (8), even if it is assigned selfishly only on the subsystem M^{S^-} of unfrozen machines, then $\forall M_j \in M^{S^-}$ it will yield induced flow $0 < t_j \leq n_j$. Therefore $\forall M_j \in M^{S^-} : t_j > 0 \Rightarrow \ell_j(t_j) \leq \ell_j(n_j) = L^N$ and by (6) we conclude that no induced selfish flow on any unfrozen link in M^{S^-} has incentive to route through any frozen machine in M^{S^+}. ∎

Lemma 7.5 *If strategy S have frozen (Def. 4.4) some (and not all) of the links it assigns flow then frozen links get no selfishly induced flow.*

Proof. Suppose that strategy S assigns on each link M_j either $s_j \geq n_j$ or even $s_j < n_j$ (freezes only some of the links it assigns flow). We show that the induced Nash T assigns $t_j = 0$ on each frozen link $M_j : s_j \geq n_j$.

N is the initial Nash equilibrium, and by Definition 4.1, $\exists\ L^N > 0$ such that for each link $M_i \in M$, if $n_i > 0$ then $\ell_i(n_i) = L^N$, otherwise $\ell_i(n_i) \geq L^N$. Let the subsystems of frozen and unfrozen links:

$$M^{S^+} = \{M_i \in M : s_i \geq n_i\} \text{ and } M^{S^-} = \{M_i \in M : s_i < n_i\}$$

Similarly as in (6), each frozen link $M_i \in M^{S^+}$ receiving induced load $t_i \geq 0$ now experiences latency

$$\ell_i^{S^+}(t_i) = \ell_i(t_i + s_i) \geq \ell_i(s_i) \geq \ell_i(n_i) \geq L^N \tag{9}$$

However, here we do not have the nice fact as in (7), since now the link latency function in M^{S^-} have been changed by S (since now may be $s_j \neq 0$ for link $M_j \in M^{S^-}$) and we can not directly apply Proposition 7.1. We can circumvent this, reworking the proof of Proposition 7.1, as follows.

Suppose that $\exists M_{i_0} \in M^{S^-} : s_{i_0} + t_{i_0} > n_{i_0}$ (thus $t_{i_0} > 0$, since $s_{i_0} < n_{i_0}$). Then $\ell_{i_0}(s_{i_0} + t_{i_0}) > \ell_{i_0}(n_{i_0})$ and since $\ell_{i_0}(n_{i_0}) \geq L^N$ we get $\ell_{i_0}(s_{i_0} + t_{i_0}) > L^N$. From this, for any link $M_j \in M^{S^-}$, it must hold: $\ell_j(s_j + t_j) > L^N$, since T is the induced equilibrium. This implies $s_j + t_j > n_j > 0$ for each link $M_j \in M^{S^-} \cap M^{N^+}$, recall (4) defining subsystems M^{N^+}, M^{N^-}. Also, for each link $M_j \in M^{S^-} \cap M^{N^-}$ we have $s_j \leq n_j = 0$ and thus $t_j \geq n_j = 0$. We conclude that if $\exists M_{i_0} \in M^{S^-} : s_{i_0} + t_{i_0} > n_{i_0}$, then (selfish & controlled) flow in unfrozen machines in M^{S^-} must be:

$$\sum_{M_i \in\ M^{S^-} \cap M^{N^+}} (s_i + t_i) + \sum_{M_i \in\ M^{S^-} \cap M^{N^-}} (s_i + t_i) > \sum_{M_i \in M^{S^-}} n_i$$

This is a contradiction, because the remaining flow than can be scheduled on all unfrozen links in M^{S^-} is:

$$r - \sum_{M_i \in M^{S^+}} s_i \ \leq \sum_{M_i \in M^{S^-}} n_i$$

since by definition of M^{S^+}, it holds

$$\sum_{M_i \in M^{S^+}} s_i \geq \sum_{M_i \in M^{S^+}} n_i$$

∎

7.4 Proof of Corollary 2.2

Here we combine the results proved in Sections 7.2 and 7.3.

First comes Theorem 7.2 (Sec. 7.2). It forces `OpTop` to freeze at least one link, otherwise `OpTop` induces equilibrium identical the initial one (a useless strategy, Def. 7.3). Now, any useful strategy must freeze all currently under-loaded (Def. 4.3) links to their optimum flow. Otherwise, Theorem 7.4 and Lemma 7.5 (Sec. 7.3), demonstrate that any non optimally frozen link will stuck to its suboptimal flow. Thus, the induced equilibrium will not yield system's optimum.

As soon as `OpTop` freezes all currently under-loaded links, it safely discards them from the system: there is no way for their flow to change. This yields a simpler subsystem.

`OpTop` recursively freezes to their optimal flows all under-loaded links and discards them from the system at hand, until it encounters a subsystem with no under-loaded links

8 Acknowledgments

We wish to thank an anonymous referee for urging us to extend `OpTop` to general nets. We also wish to thank Ioannis Chatzigiannakis for useful ideas.

References

[1] Network classics. Technical report. http://supernet.som.umass.edu/classics.htm.

[2] T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, London and San Diego, 1995.

[3] V. Bonifaci, T. Harks, and G. Schaefer. Stackelberg routing in arbitrary networks. In *4th International Workshop On Internet And Network Economics (WINE)*, 2008.

[4] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: motivation, formulation, and example. *IEEE/ACM Trans. Netw.*, 1(6):614–627, 1993.

[5] J. Correa and N. Stier-Moses. Stackelberg routing in atomic network games. In *Submitted*.

[6] A. Czumaj. *Selfish routing on the Internet*. In Handbook of Scheduling. CRC Press, 2004.

[7] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 413–420, 2002.

[8] S. Dafermos and F. Sparrow. The Traffic Assignment Problem for a General Network. *Research of the National Bureau of Standards*, 73B:91–118, 1969.

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

[10] *Dijkstra's algorithm*. http://en.wikipedia.org/wiki/`Dijkstra's_algorithm`.

[11] P. Dubey. Inefficiency of nash equilibria. *Math. Oper. Res.*, 11(1):1–8, 1986.

[12] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *J. Comput. Syst. Sci.*, 63(1):21–41, 2001.

[13] D. Fotakis. Stackelberg strategies for atomic congestion games. In *15th European Symposium on Algorithms (ESA)*, pages 299–310, 2007.

[14] D. Fotakis, S. C. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. G. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. In *29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 123–134, 2002.

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

[16] A. Kaporis and P. Spirakis. The price of optimum in stackelberg games on arbitrary single commodity networks and latency functions. In *18th annual ACM symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 19–28, 2006.

[17] A. Kaporis and P. Spirakis. *Selfish splittable flows and NP-completeness.* Handbook NP-Completeness: Theory and Applications. Chapman & Hall, CRC, in preparation.

[18] A. C. Kaporis, E. Politopoulou, and P. G. Spirakis. The price of optimum in stackelberg games. *Electronic Colloquium on Computational Complexity (ECCC)*, (056), 2005.

[19] G. Karakostas and S. G. Kolliopoulos. Stackelberg strategies for selfish routing in general multicommodity networks. Technical report, AdvOL-Report 2006/08. Also in *ALGORITHMICA*, 2007.

[20] Y. Korilis, A. Lazar, and A. Orda. The designer's perspective to noncooperative networks. In *IEEE INFOCOM*, 1995.

[21] Y. A. Korilis, A. A. Lazar, and A. Orda. Achieving network optima using stackelberg routing strategies. *IEEE/ACM Trans. Netw.*, 5(1):161–173, 1997.

[22] Y. A. Korilis, A. A. Lazar, and A. Orda. Capacity allocation under noncooperative routing. *IEEE Transactions on Automatic Control*, 42:161–173, 1997.

[23] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, 1563:404–413, 1999.

[24] V. S. A. Kumar and M. V. Marathe. Improved results for stackelberg scheduling strategies. In *29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 776–787, 2002.

[25] H. Lin, T. Roughgarden, and E. Tardos. A stronger bound on braess's paradox. In *15th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 340–341, 2004.

[26] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *33th Annual ACM Symposium on Theory of Computing (STOC)*, pages 510–519, 2001.

[27] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[28] N. Nisan. Algorithms for selfish agents: Mechanism design for distributed computation. In *16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 1–15, 1999.

[29] N. Nisan and A. Ronen. Algorithmic mechanism design. In *31st Annual Symposium on Theory of Computing (STOC)*, pages 129–140. ACM, 1999.

[30] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[31] M. Osborne and A. Rubinstein. *A course in Game Theory*. MIT Press.

[32] G. Owen. *Game Theory*. Academic Press, 3rd edition, 1995.

[33] C. Papadimitriou. Algorithms, games, and the internet. In *33rd Annual Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

[34] C. Papadimitriou. Game theory and mathematical economics: A theoratical computer scientist's introduction. In *42nd IEEE Annual Symposium of Foundations of Computer Science (FOCS)*, 2001.

[35] M. Patriksson. *The traffic assignment problem - models and methods*. Linköping Institute of Technology, Sweden, 1991.

[36] T. Roughgarden. Designing networks for selfish users is hard. In *42nd IEEE Annual Symposium of Foundations of Computer Science (FOCS)*, pages 472–481, 2001.

[37] T. Roughgarden. Slides presentation of "*Stackelberg scheduling strategies*". In *STOC*, 2001. http://theory.stanford.edu/∼tim/slides/`stack_cornell.pdf`.

[38] T. Roughgarden. Stackelberg scheduling strategies. In *33rd Annual Symposium on Theory of Computing (STOC)*, pages 104–113, 2001.

[39] T. Roughgarden. The price of anarchy is independent of the network topology. In *34th Annual Symposium on Theory of Computing (STOC)*, pages 428–437, 2002.

[40] T. Roughgarden. *Selfish Routing*. Ph.D dissertation, Cornell University, USA, May 2002. http://theory.stanford.edu/∼tim/.

[41] T. Roughgarden. Stackelberg scheduling strategies. *SIAM J. Comput.*, 33(2):332–350, 2004.

[42] T. Roughgarden. *Selfish Routing and the Price of Anarchy*. MIT Press, Cambridge, MA, USA, 2005.

[43] T. Roughgarden and É. Tardos. How bad is selfish routing? In *41st IEEE Annual Symposium of Foundations of Computer Science (FOCS)*, pages 93–102, 2000.

[44] Y. Sharma and D. Williamson. Stackelberg thresholds in network routing games or the value of altruism. In *8th ACM conference on Electronic Commerce (EC)*, 2007.

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

[46] C. Swamy. The effectiveness of stackelberg strategies and tolls for network congestion games. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[47] H. von Stackelberg. *Marktform und Gleichgewicht*. Springer-Verlag, 1934.