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
26500 Patras, Greece
kaporis@ceid.upatras.gr

P.G. Spirakis[†]
Research Academic Computer Technology Institute and University of Patras
P.O. Box 1122 Patras, 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* [12, 26]. 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, α), $0 \le \alpha \le 1$. It was shown [23] 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 needed by a Leader to induce M's optimum cost, as well as his optimal strategy.

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 $\le \frac{1}{\alpha}$ 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.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

General Terms

Algorithms, Economics

[*]Partially supported by European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the program *PYTHAGORAS I*

[†]Partially supported by the IST EU Program Delis IST-2004-001907.

Keywords

Stackelberg, Price of Optimum, Coordination Ratio

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 [6]. Such *selfish* behavior is being studied with the notion of *Nash Equilibrium* in the mathematical framework of Game Theory [5, 10, 11, 12, 15, 18, 19, 23, 27].

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 [12]. This work has been extended (*price of anarchy* is another equivalent term) in [4, 3, 8, 14, 21, 20, 24, 23, 26, 22].

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 [2], algorithmic mechanisms [7, 17, 16], network design [9, 22], or routing small portion of the traffic centrally [13, 10, 23].

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 [13, 1, 10, 11, 23, 29]. 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

(i) Single-commodity networks with parallel links. Consider a system M of parallel links and a total of flow r 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 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 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 [26], but if all links in M have linear load depended latency functions, then $\epsilon_{(M,r)} \leq 4/3$ [12]. 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 [23], and as in [10] where we focus on the case of an infinite number of users. According to [23, 10] there is a central authority (*Leader*) that controls a portion $0 \leq \alpha \leq 1$ of the overall flow r 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 [23] 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 cost depends on the strategy chosen by the Leader and on the portion of the flow that she controls. More precisely, in [23] 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 $\epsilon_{(M,r,\alpha)} \leq \frac{4}{3+\alpha}$. 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 [13]. More precisely, [23] 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 [23]. 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 [23] stating that the problem of computing the Optimal Stackelberg strategy on a given Stackelberg scheduling instance (M, r, α) is weakly NP-hard.

(ii) Arbitrary single-commodity nets. Finally, an important question of [23] 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 his αr portion on some edges, inducing a selfish $s \rightarrow t$ routing of the remaining flow with best possible cost? In [25] Appendix B.3 Proposition B.3.1, 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*. To the best of our knowledge, no performance guarantee as a function of the centrally controlled portion α has been established for arbitrary (s, t)-networks.

(iii) Arbitrary multi-commodity nets. Even less is known for Stackelberg strategies on arbitrary networks.

1.2 Our results

(i) Single-commodity network with parallel links. Our first main result is the polynomial-time algorithm `OpTop` that on input a scheduling instance (M, r) computes the minimum portion β_M of flow r needed by a Leader to induce the Optimum cost $C(O)$ on M, as well as the Leader's Optimal Stackelberg strategy (see the open question in [28] page 28). In other words, for an arbitrary scheduling instance (M, r) and arbitrary continuous, differentiable and strictly increasing latency functions we prove that for all Stackelberg scheduling instances of the form $(M, r, \alpha \geq \beta_M)$ a Leader can enforce the Optimum cost $C(O)$ on M, and the problem of computing the Optimum Stackelberg strategy is in P. 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 as the ones presented in [13]. Such non-optimizing behavior was presented also in [10], 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.

Single-commodity network with parallel links, focusing on the hard region. Trying to understand further the underlying complexity of hard instances $(M, r, \alpha < \beta_M)$, we started to investigate systems of 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 [10]. We compute the optimal Stackelberg strategy on hard instances $(M, r, \alpha < \beta_M)$ for any network with parallel links $M = \{M_1, \ldots, M_m\}$ where each link $M_i \in M$ has linear latency $\ell_i(x) = a_i x + b_i$ satisfying the property: $b_1 \leq \ldots \leq b_m$ and $a_1 = \ldots = a_m$.

(ii) Arbitrary single-commodity nets. Our second main result is an algorithm that efficiently computes the minimum portion β_G of Leader's flow, sufficient to induce the optimum routing of flow r from a source-vertex s to a sink t on *any* network G. Despite the negative results presented in [23, 25], we *can* modify `OpTop` to work on input an arbitrary network G. Our algorithm also computes the associated optimal strategy of the Leader.

(iii) Arbitrary multi-commodity nets. We conjecture that our results also hold for k commodities on arbitrary nets.

1.3 Outline of the paper

In Section 1.4 we describe the model that we use for the case of parallel links and the corresponding one for arbitrary networks. In Section 2 we present the *polynomial-time* algorithm OpTop for parallel links. In Section 2.2 we prove its optimality for parallel links. In Section 2.3 we give a slightly different algorithm for arbitrary networks. We take care of the Braess graph as an example. We present our results in this order for reasons of clarity. In Section 3 we focus on the hard region $(M, r, \alpha < \beta_M)$ of parallel links.

1.4 The Model and the problem

Single source-destination parallel links: We have m parallel links connecting a single source vertex s to a sink vertex t. Each link i on flow x_i incurs latency function $\ell_i(x_i) \geq 0$, differentiable, strictly increasing and $x_i\ell_i(x_i)$ convex on x_i. Each selfish user controls an infinitesimally small amount of the total flow r. Let the m-vector $X \in \mathcal{R}_+^m$ denote the assignment of jobs to the links in M such that $\sum_{i=1}^m x_i = r$. This instance is annotated (M, r). The *Cost* of an assignment $X \in \mathcal{R}_+^m$ on the (M, r) instance is $C(X) = \sum_{i=1}^m x_i\ell_i(x_i)$ The minimum cost is incurred by a unique assignment $O \in \mathcal{R}_+^m$, called the *Optimum* assignment. The unique assignment $N \in \mathcal{R}_+^m$ defines a *Nash equilibrium*, if no user can find a loaded machine with $<$ latency than any other machine.

Arbitrary Networks: A network is as a directed graph $G(V, E)$ with set of vertices V and edges E. There are k source-destination vertex pairs $(s_1, t_1), \dots, (s_k, t_k)$ and no self loops are allowed. $\mathcal{P}_i$ is the set of all paths amongst $(s_i, t_i), i = 1, \dots, k$, and $\mathcal{P} = \bigcup_i \mathcal{P}_i$. A flow is a function $f : \mathcal{P} \rightarrow R^+$. The amount of flow f_e on edge $e \in E$ is the flow it receives from all paths in $\mathcal{P}$. If we focus on the flow of a specific source-destination pair (s_i, t_i) then we let f^i the restriction of f to $\mathcal{P}_i, i = 1, \dots, k$. 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 each path $\mathcal{P}_i$ is r_i. If flow f_e traverses edge e it incurs latency $\ell_e(x_e)$, where $\ell_e(\cdot)$ is increasing, differentiable and $f_e\ell_e(f_e)$ convex on f_e. 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 f^* is the one minimizing the cost $C(\cdot)$ of scheduling flow r on graph G and due to convexity properties can be efficiently computed. We have a Nash equilibrium on a network G if an only if for every commodity $i \in \{1, \dots, k\}$ 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 in a loaded path can find any other path from s_i to t_i experiencing less latency.

Problem: The input is a scheduling instance (M, r) where M is either m parallel links or an $s{-}t$ network. The question is to compute efficiently the *minimum* portion of the flow controlled by the Leader (and *the associated strategy* of the Leader) to induce the overall optimum cost on M.

2. OUR ALGORITHM AND ITS OPTIMALITY

Single-commodity network with parallel links. In this section we present algorithm OpTop that computes the *minimum* flow that should be controlled by the Leader in order to induce the overall optimum on a given instance (M, r). Let $O := \langle o_1, \dots, o_m \rangle$ the optimum assignment and $N := \langle n_1, \dots, n_m \rangle$ the Nash assignment on (M, r). Intuitively[1], OpTop initially loads $s_i = o_i$ to each link $M_i \in M$ with $n_i < o_i$, that is, to all links not appealing to the selfish users. Then it discards all these not appealing links as soon as it loads them optimally. The remaining flow is assigned recursively by OpTop in exactly the same fashion to the simplified subnetwork of links. It terminates as soon as it encounters a simplified subnetwork with all of its links optimally loaded.

```
Algorithm: OpTop (Parallel links)
(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
    $\beta_M = (r_0 - r)/r_0$.
```

In Sections 2.1 and 2.2 we prove our main Theorem 1 for the case of a single-source, single-destination network M of parallel links.

THEOREM 1. *Consider an instance (M, r) with latency function $\ell_i(\cdot)$ per link $M_i \in M$ differentiable, strictly increasing and $x\ell_i(x)$ convex on x. Algorithm* OpTop *computes the* minimum *portion β_M of total flow r that a Leader must control to induce overall optimum cost on M, as well as Leader's optimal Stackelberg strategy.*

In Section 2.3 we generalize Theorem 1 for arbitrary network topologies and the same standard class of latency functions.

2.1 Useful machinery

Single-commodity network with parallel links. We denote the corresponding *Nash* and *Optimum* assignments on an instance (M, r) as $N = \langle n_1, \dots, n_m \rangle$ with $\sum_{i=1}^m n_i = r$, and $O = \langle o_1, \dots, o_m \rangle$ with $\sum_{i=1}^m o_i = r$. We give a more useful definition for the Nash assignment N.

Definition 1. An assignment $N = \langle n_1, \dots, n_m \rangle$of total flow $\sum_{i=1}^m n_i = r$ on (M, r) is called *Nash Equilibrium* if there exists a constant $L^N > 0$ such that for each link $M_i \in M$, if $n_i > 0$ then $\ell_i(n_i) = L^N$, otherwise $\ell_i(n_i) \geq L^N$.

We denote as *Stackelberg* strategy S an assignment $S = \langle s_1, \dots, s_m \rangle$ of flow $\sum_{i=1}^m s_i = \beta r$, $\beta \in [0,\ 1]$, on instance (M, r). Given S, we denote the *induced Nash* assignment as $T = \langle t_1, \dots, t_m \rangle$ with $\sum_{i=1}^m t_i = (1 - \beta)r$.

Definition 2. Given Stackelberg strategy $S = \langle s_1, \dots, s_m \rangle$ with $\sum_{i=1}^m s_i = \beta r$ and $\beta \in [0,\ 1]$, the assignment $T =$

[1] OpTop finds all shortest paths (fast edges) with respect to O and freezes the flow on all slow edges, see Section 2.3. Here our proof methodology identifies recursively the slow-edges, while in Section 2.3 we adopt a non-recursive argument.

$\langle t_1,\ldots,t_m\rangle$ of the remaining flow $\sum_{i=1}^m t_i=(1-\beta)r$ on instance (M,r) is an *Induced Nash Equilibrium* if there exists a constant $L^S>0$ such that for each $M_i\in M$, if $t_i>0$ then $\ell_i(t_i+s_i)=L^S$, otherwise $\ell_i(t_i+s_i)=\ell_i(0+s_i)\geq L^S$.

The Stackelberg Stategy S induces the Nash assignment T. The assignment $T+S$ is called *Stackelberg Equilibrium*. The *Cost* of an assignment $X=\langle x_1,\ldots,x_m\rangle$ on M equals $C(X)=\sum_{i=1}^m x_i\ell_i(x_i)$. The cost of the Stackelberg Equilibrium $S+T$ is $C(S+T)=\sum_{i=1}^m(s_i+t_i)\ell_i(s_i+t_i)$.

Definition 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. Link $M_i\in M$ (or load $s_i\in S$) is called *frozen* if Stackelberg strategy S assigns to it load $s_i\geq n_i,\ i=1,\ldots,m$.

Luckily, by the Nash assignment N of the users, all links may end up optimum-loaded. 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 $C(S+T)=C(O)$ of system M. It is convenient to state the following easy proposition.

PROPOSITION 1. *Consider an instance (M,r) with latency functions $\ell_j(\cdot)$ $j=1,\ldots,m$. Let the Nash assignment $N=\langle n_1,\ldots,n_m\rangle$ of (M,r). If $N'=\langle n'_1,\ldots,n'_m\rangle$ is the Nash assignment of (M,r') with $r'\leq r$, then for each link $M_i\in M$ it holds: $n'_i\leq n_i$.*

PROOF. Since N is a Nash assignment of the flow r on M, by Definition 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 $M^{N^+}=\{M_i\in M: n_i>0\}$ and $M^{N^-}=\{M_i\in M: n_i=0\}$. Similarly for N', let $L^{N'}>0$ the corresponding constant, and $M^{N'^+}=\{M_i\in M: n'_i>0\}$ and $M^{N'^-}=\{M_i\in M: n'_i=0\}$. To reach a contradiction, suppose that $\exists\ M_{i_0}\in M^{N'^+}$ such that $n'_{i_0}>n_{i_0}$.

Case 1: If $M_{i_0}\in M^{N^-}$ then $\ell_{i_0}(n'_{i_0})=L^{N'}>\ell_{i_0}(n_{i_0})=\ell_{i_0}(0)\geq L^N$, since each $\ell_i(\cdot)$ is strictly increasing and $n'_{i_0}>n_{i_0}=0$. 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 $r'\geq\sum_{M_i\in M^{N^+}}n'_i>\sum_{M_i\in M^{N^+}}n_i=r$.

Case 2: If $M_{i_0}\in M^{N^+}$ then $\ell_{i_0}(n'_{i_0})=L^{N'}>\ell_{i_0}(n_{i_0})=L^N$. Therefore, each link $M_i\in M^{N^+}$ must receive load $n'_i>n_i$ under N', otherwise each $M_i\in M^{N^+}$ will experience latency $\ell_i(n'_i)\leq\ell_i(n_i)=L^N<L^{N'}$. That is, in the same fashion, we reach a contradiction. $\square$

Theorem 2 describes each Stackelberg strategy S inducing Nash assignment T with cost $C(S+T)=C(N)$. In other words, Theorem 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 2. *Consider an instance (M,r) with latency functions $\ell_j(\cdot)$ $j=1,\ldots,m$. Let the Nash assignment $N=\langle n_1,\ldots,n_m\rangle$ of (M,r). Suppose that for a Stackelberg strategy $S=\langle s_1,\ldots,s_m\rangle$ with $\sum_{i=1}^m s_i=\beta r,\beta\in[0,\ 1]$, it holds $s_j\leq n_j,\ j=1,\ldots,m$. Given S, let the induced Nash assignment $T=\langle t_1,\ldots,t_m\rangle$ of the remaining flow $(1-\beta)r$. Then it holds $n_j=s_j+t_j$ for each $M_j\in M$, in other words, $N\equiv S+T$.*

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 $\sum_{i=1}^m s_i=\beta r,\beta\in[0,\ 1]$. Then, the initial system of links M is transformed by S to the equivalent system M^S, such that each link $M_j^S\in M^S$ with load x_j now experiences latency $\ell_j^S(x_j)=\ell_j(x_j+s_j)$. 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 $T=\langle t_1,\ldots,t_m\rangle$ this assignment on M^S. Obviously, for the same constant $L^N>0$ as above, it holds: $\ell_j^S(t_j)=\ell_j\left((n_j-s_j)+s_j\right)=\ell_j(n_j)=L^N$, for each $M_j\in M$ with $t_j>0$. This means that T is a Nash equilibrium on system M^S and also $S+T\equiv N$. $\square$

In view of the negative result of Theorem 2, a natural question concerns the properties that a Stackelberg strategy must have in order to induce cost $\neq C(N)$. We answer this question on Theorem 3 and its generalization Lemma 1 below. Before this, we give a convenient definition.

Definition 5. Each Stackelberg strategy S that satisfies Theorem 2 is called *useless-strategy*, otherwise is called *useful-strategy.*

Theorem 3 states that any link $M_i\in M$ receiving load $s_i\geq n_i$ by a strategy S (while there is no link $M_j\in M$ with load $s_j<n_j$) will become non appealing for the subsequent selfish assignment T of the users. That is, for each $M_i\in M$ assigned load $s_i\geq n_i$, its induced load by the Nash assignment T equals $t_i=0$. Intuitively, in the induced Nash equilibrium T, the dictated load $s_i\geq n_i$ by strategy S to link M_i will remain "*frozen*" to $s_i,\ i\in\{1,\ldots,m\}$.

THEOREM 3. *Let the Nash assignment $N=\langle n_1,\ldots,n_m\rangle$ of on instance (M,r). Suppose that for a Stackelberg strategy $S=\langle s_1,\ldots,s_m\rangle$ with $\sum_{i=1}^m s_i=\beta r$ we have either $s_j\geq n_j$ or $s_j=0, j=1,\ldots,m$. Then for the induced Nash assignment $T=\langle t_1,\ldots,t_m\rangle$ of the remaining load $(1-\beta)r$ we have that $t_j=0$ for each $M_j\in M$ such that $s_j\geq n_j,\ j=1,\ldots,m$.*

PROOF. By Definition 1, since N is a Nash equilibrium on M, $\exists\ L^N>0$ such that for each $M_i\in M$, if $n_i>0$ then $\ell_i(n_i)=L^N$, otherwise $\ell_i(n_i)\geq L^N$. Fix a Stackelberg strategy S on M, such that for each link $M_i\in M$, either

$s_i \geq n_i$ or $s_i = 0$. Let $M^{S^+} = \{M_i \in M : s_i \geq n_i\}$ and $M^{S^-} = \{M_i \in M : s_i = 0\}$, and notice that $M = M^{S^+} \cup M^{S^-}$ and $M^{S^+} \cap M^{S^-} = \emptyset$. Each $M_i \in M^{S^+}$ receiving induced load $t_i \geq 0$ now experiences latency

$$\ell_i^{S^+}(t_i) = \ell_i(t_i + s_i) \geq \ell_i(s_i) \geq \ell_i(n_i) \geq L^N. \tag{3}$$

On the other hand, each $M_j \in M^{S^-}$ receiving induced load $t_j \geq 0$ experiences the *same* (since $s_j = 0$) as the *initial* (that is, without applying strategy S) latency

$$\ell_j^{S^-}(t_j) = \ell_j(t_j + s_j) = \ell_j(t_j). \tag{4}$$

In the sequel, the induced Nash assignment T by strategy S assigns the remaining flow on M

$$r - \sum_{i=1}^{m} s_i = r - \sum_{M_j \in M^{S^+}} s_j \leq \sum_{M_j \in M^{S^-}} n_j. \tag{5}$$

Having in mind (4) and (5), the crucial observation is that *even* if all the remaining flow that appears in LHS of (5) is assigned selfishly *only* on subsystem M^{S^-}, it is impossible the common latency L^{S^-} experienced by each loaded link in M^{S^-} to become $L^{S^-} > L^N$, so that at least one link in M^{S^+} to become appealing for the selfish players. More formally, let T^{S^-} be the *partial* Nash assignment that corresponds to assigning the flow that appears in LHS of (5) *only* on to the subsystem M^{S^-}. By Definition 1, $\exists\ L^{S^-} > 0$ such that for each loaded link $M_j \in M^{S^-}$ with load $0 < t_j^{S^-} \leq n_j$ (here RHS inequality stems from Proposition 1 and the RHS of (5)) it holds

$$\ell_j^{S^-}(t_j^{S^-}) = \ell_j(t_j^{S^-}) = L^{S^-} \leq \ell_j(n_j) = L^N. \tag{6}$$

By (3) and (6) it follows that no link $M_j \in M^{S^+}$ is appealing for the *overall* induced Nash assignment T. $\square$

In view of the negative result of Theorem 2, a natural question concerns the properties that a Stackelberg strategy must have in order to induce cost $\neq C(N)$. Lemma 1 rules out such possibility. Intuitively, Lemma 1 states that each assignment of load $s_j \geq n_j$ made by strategy S to each link $M_j \in M^{S^+}$ (see its definition in Proposition 2 below, it is the subset of frozen links) remains *unaffected* by its induced Nash load (i.e. T induces load $t_j = 0$ on M_j), *irrespectively* of any assignment of load $s_i < n_i$ made by S to any other link $M_i \neq M_j$.

LEMMA 1. *Let the Nash assignment $N = \langle n_1, \ldots, n_m\rangle$ on instance (M, r). Suppose that for a Stackelberg strategy $S = \langle s_1, \ldots, s_m\rangle$ with $\sum_{i=1}^m s_i = \beta r, \beta \in [0,\ 1]$, we have either $s_j \geq n_j$ or $s_j < n_j, j = 1, \ldots, m$. Then for the induced Nash assignment $T = \langle t_1, \ldots, t_m\rangle$ of the remaining load $(1-\beta)r$ we have that $t_j = 0$ for each link $M_j \in M$ such that $s_j \geq n_j,\ j = 1, \ldots, m$.*

PROOF. By Definition 1, since N is a Nash equilibrium on M, $\exists\ L^N > 0$ such that for each link $M_i \in M$, if $n_i > 0$ then $\ell_i(n_i) = L^N$, otherwise $\ell_i(n_i) \geq L^N$. Consider an arbitrary strategy S and let $M^{S^+} = \{M_i \in M : s_i \geq n_i\}$ and $M^{S^-} = \{M_i \in M : s_i < n_i\}$. Similarly as in (3), each $M_i \in M^{S^+}$ receiving induced load $t_i \geq 0$ now experiences latency

$$\ell_i^{S^+}(t_i) = \ell_i(t_i + s_i) \geq \ell_i(s_i) \geq \ell_i(n_i) \geq L^N. \tag{7}$$

However, here we do not have the nice fact as in (4) for the link latencies in M^{S^-} (since now $s_j \neq 0$). We can circumvent this as follows. The induced Nash assignment T assigns on system M the remaining flow that equals

$$\begin{aligned} r^S &= r - \sum_{M_i \in\ M^{S^+} \cup M^{S^-}}^{m} s_i \\ &\leq r^{S^-} = r - \sum_{M_i \in M^{S^+}} s_i \\ &\leq \sum_{M_i \in M^{S^-}} n_i, \end{aligned} \tag{8}$$

where the rightmost inequality stems from the fact that

$$r^{S^+} = \sum_{M_i \in M^{S^+}} s_i \geq \sum_{M_i \in M^{S^+}} n_i. \tag{9}$$

Now, we prove that *even* if the flow r^{S^-} in (8) is scheduled selfishly *only* on the subsystem M^{S^-}, then all links with load > 0 in it, would not experience common latency $L^{S^-} > L^N$ so that at least one link in M^{S^+} to become appealing for scheduling any excess of flow. Let N^{S^-} the *partial* Nash assignment (that is, without previously assigning S on to subsystem M^{S^-}) when scheduling flow r^{S^-} appearing in (8) *only* on to subsystem M^{S^-}. Also, applying strategy S on to subsystem M^{S^-}, let T^{S^-} the *induced partial* Nash assignment (that is, by assigning previously S on to subsystem M^{S^-}) when scheduling the remaining of r^{S^-} appearing in (8) *only* on to subsystem M^{S^-}. Let $n_i^{S^-}$ (or $t_i^{S^-}$) denote the load assigned by N^{S^-} (or T^{S^-}) to each $M_i \in M^{S^-}$. Then, we have the following two cases.

Case 1: Suppose that for each link $M_i \in M^{S^-}$ it holds $s_i \leq n_i^{S^-}$. Then we can apply Theorem 2 when assigning the remaining of r^{S^-} on subsystem M^{S^-}. In this way, for each link $M_i \in M^{S^-}$ it holds $s_i + t_i^{S^-} = n_i^{S^-}$. Furthermore, from Inequality (8) we realize that

$$r^{S^-} = \sum_{M_i \in M^{S^-}} n_i^{S^-} \leq \sum_{M_i \in M^{S^-}} n_i. \tag{10}$$

Applying Proposition 1, we conclude that for each loaded link $M_i \in M^{S^-}$ it holds $\ell_i(n_i^{S^-}) \leq \ell_i(n_i) = L^N$ and using (7) the lemma is proved.

Case 2: Suppose that there exists at least one link $M_i \in M^{S^-}$ such that $s_i > n_i^{S^-}$. Then we can construct T^{S^-} as follows. For each link $M_i \in M^{S^-}$ such that $s_i \leq n_i^{S^-}$ we set

$$t_i^{S^-} \leq n_i^{S^-} - s_i \tag{11}$$

(see its validity below) otherwise we set $t_i^{S^-} = 0$. Clearly, each link $M_i \in M^{S^-}$ with $t_i^{S^-} > 0$ experiences a common latency

$$\begin{aligned} L^{S^-} &= \ell_i^{S^-}(t_i^{S^-}) \\ &\leq \ell_i((n_i^{S^-} - s_i) + s_i) = \ell_i(n_i^{S^-}) \\ &\leq \ell_i(n_i) = L^N, \end{aligned} \tag{12}$$

and the lemma follows. On the other hand, each link $M_i \in M^{S^-}$ with $t_i^{S^-} = 0$ already has load s_j due to S such that $n_j^{S^-} < s_j < n_j$. Therefore,

$$L^{S^-} \leq \ell_j^{S^-}(t_j^{S^-}) = \ell_j^{S^-}(0) = \ell_j(s_j) < \ell_j(n_j) = L^N,$$

and the lemma follows. The Inequality (11) can be proved as follows. Given strategy S, let the subsystem $M_0^{S^-} \subseteq M^{S^-}$ containing all links $M_i \in M^{S^-}$ such that $s_i > n_i^{S^-}$. Consider strategy S' such that on each $M_i \in M^{S^-} \setminus M_0^{S^-}$ it assigns load $s'_i = s_i$ and for each $M_0^{S^-}$ it assigns load $s'_i = s_i - (s_i - n_i^{S^-}) = n_i^{S^-}$ (that is, it subtracts load $(s_i - n_i^{S^-})$). In this way we get

$$\sum_{M_i \in M^{S^-}} s'_i < \sum_{M_i \in M^{S^-}} s_i. \tag{13}$$

Given strategy S', Theorem 2 applies on assigning selfishly the flow that appears on the RHS of (14) onto subsystem M^{S^-}, and let $T^{S'^-}$ the corresponding induced Nash assignment. Then, for each link $M_i \in M^{S^-}$ it holds $t_i^{S'^-} = n_i^{S^-} - s'_i$, and *most* importantly, each $M_i \in M_0^{S^-}$ gets induced load $t_i^{S'^-} = 0$. The crucial observation is that if we *add back* the subtracted load $(s_i - n_i^{S^-})$ on each $M_i \in M_0^{S^-}$ then (i) strategy S' *becomes* S and (ii) each $M_i \in M_0^{S^-}$ becomes *even* less appealing (recall $t_i^{S'^-} = 0$ under S'). Furthermore, given strategy S, let T^{S^-} the induced Nash assignment of the flow that appears in the LHS of (14)

$$\begin{aligned} \sum_{M_i \in M^{S^-}} t_i^{S^-} &= r^{S^-} - \sum_{M_i \in M^{S^-}} s_i \\ &< \sum_{M_i \in M^{S^-}} t_i^{S'^-} \\ &= r^{S^-} - \sum_{M_i \in M^{S^-}} s'_i, \end{aligned} \tag{14}$$

on subsystem M^{S^-}. From Proposition 1, on selfisly assigning the flow in LHS of (14) onto subsystem $M^{S^-} \setminus M_0^{S^-}$, we conclude that each $M_i \in M^{S^-} \setminus M_0^{S^-}$ now receives flow

$$t_i^{S^-} \leq t_i^{S'^-} = n_i^{S^-} - s'_i = n_i^{S^-} - s_i,$$

while each $M_i \in M_0^{S^-}$ now receives $t_i^{S^-} = 0$. □

In Section 2.2 we apply Theorem 3, Lemma 1 and Proposition 2 to discard the links with frozen load $s_j = o_j \geq n_j$ and simplify the initial game. Clearly, such links will never be affected by the induced selfish play of the users on the rest of links. Therefore, using Proposition 2, we focus on the remaining links with load under S that equals $s_i < n_i$, which may be affected by the selfish users, trying to find a subsequent partial Stackelberg strategy on them that will induce the optimum cost.

PROPOSITION 2. *Let the system $M = \{M_1, \ldots, M_m\}$ and the Nash assignment $N = \langle n_1, \ldots, n_m \rangle$ on (M, r). Fix a Stackelberg strategy $S = \langle s_1, \ldots, s_m \rangle$ such that either $s_j \geq n_j$ or $s_j < n_j, j = 1, \ldots, m$. Let the subset of frozen links $M^{S^+} = \{M_j \in M : s_j \geq n_j\}$, $j = 1, \ldots, m$, and their frozen load $r^{S^+} = \sum_{M_j \in M^{S^+}} s_j$. Then the initial Stackelberg game of flow r on M can be simplified to scheduling the remaining unfrozen flow $r^{S^-} = r - r^{S^+}$ to the remaining unfrozen subsystem of links $M^{S^-} = M \setminus M^{S^+}$.*

2.2 The optimal evolution of `OpTop`

2.2.1 Phase 1: `OpTop` *loads optimally all initially under-loaded links.*

During PHASE $i \geq 1$, let $N^i = \langle n_1^i, \ldots, n_m^i \rangle$ denote the Nash assignment of flow r^i (where r^1 equals the initial total flow r) on to subsystem of links M^i (where M^1 is the initial system M). Also, let $O = \langle o_1, \ldots, o_m \rangle$ denote the overall optimum assignment of flow r onto system M (notice that O is not parameterized with respect to the ith PHASE). We introduce the following partition (according to Definition 3) of the system $M^1 \equiv M$ of links, during PHASE 1

$$\begin{aligned} M^{1^-} &= \{M_j \in M^1 : n_j^1 < o_j\} \\ M^{1^+} &= \{M_j \in M^1 : n_j^1 \geq o_j\}. \end{aligned} \tag{15}$$

According to Theorem 3 and Lemma 1, if during PHASE 1 a Stackelberg strategy $S^1 = \langle s_1^1, \ldots, s_m^1 \rangle$ assigns load s_j^1 such that $o_j < n_j^1 < s_j^1$ to at least one over-loaded link $M_j \in M^{1^+}$ (see Definition 3) then M_j will remain frozen to an unfavorably high value $s_j^1 > o_j$, irrespectively of any load s_i^1 strategy S^1 may assign to any other link $M_i \neq M_j$. Therefore, M_j will never reduce its load to the optimum value o_j, and thus the system M will never converge to its overall optimum assignment O.

In the same fashion, applying Lemma 1, if during PHASE 1 strategy S^1 assigns load s_j^1 such that $n_j^1 < s_j^1 < o_j$ to at least one under-loaded link $M_j \in M^{1^-}$ then M_j will remain frozen to an unfavorably low load $s_j^1 < o_j$.

Then `OpTop` must assign load $s_j^1 = o_j > n_j^1$ to each initially under-loaded link $M_j \in M^{1^-}$, otherwise under-loaded links will never attain their overall optimum load. Furthermore, by Theorems 2, 3 and Lemma 1, it is wasteful any assignment of flow $s_i^1 < n_i^1$ to any other link $M_i \in M^1$. Clearly, no such assignment $o_i < s_i^1 < n_i^1$ can affect favorably any load assignment $s_j^1 = o_j > n_j^1$ to any initially under-loaded link $M_j \in M^{1^-}$. We conclude that at the end of PHASE 1 algorithm `OpTop` constructs the Stackelberg strategy S^1 such that in each $M_j \in M^{1^-}$ it assigns load $s_j^1 = o_j$, while in each $M_j \in M^{1^+}$ it assigns $s_j^1 = 0$.

Simplification of the initial game: Each initially under-loaded link $M_j \in M^{1^-}$ will remain frozen to its induced by S^1 optimum load $s_j^1 = o_j > n_j^1$. Applying Proposition 2, we can simplify the game by discarding each initially under-loaded link $M_j \in M^{1^-}$ that becomes frozen by S^1. During PHASE 1 algorithm `OpTop` needs a portion r^{1^-} to frozen the links in M^{1^-} $r^{1^-} = \sum_{M_j \in M^{1^-}} o_j$. Then the initial Stackelberg game of flow r^1 on M^1 can be simplified to scheduling the remaining flow $r^2 = r^1 - r^{1^-}$ to the remaining links $M^2 = M^1 \setminus M^{1^-}$.

2.2.2 *Phase $i \geq 2$: the recursive nature of* `OpTop`.

During Phase 2, we consider the *Nash* assignment $N^2 = \langle n_j^2 : M_j \in M^2 \rangle$ when scheduling the remaining flow $r^2 = r^1 - r^{1^-}$ on the simplified system $M^2 = M^1 \setminus M^{1^-}$ and, similarly as in Phase 1 let,

$$\begin{aligned} M^{2^-} &= \{M_j \in M^2 : \ n_j^2 < o_j\} \\ M^{2^+} &= \{M_j \in M^2 : \ n_j^2 \geq o_j\}. \end{aligned} \qquad (16)$$

Then, applying similarly as in Section 2.2.1 Theorem 2 and 3, Lemma 1 and Proposition 1, `OpTop` constructs the subsequent *Stackelberg* strategy S^2 onto subsystem M^2 such that in each $M_j \in M^{2^-}$ it assigns $s_j^2 = o_j$, while in each $M_j \in M^{2^+}$ it assigns $s_j^2 = 0$. Then, once more, `OpTop` simplifies the game, scheduling flow $r^3 = r^2 - r^{2^-} = r^2 - \sum_{M_j \in M^{2^-}} o_j$ onto subsystem $M^3 = M^2 \setminus M^{2^-}$. Finally, `OpTop` terminates as soon as it reaches a Phase i_0 where the simplified subsystem M^{i_0} has the property

$$M^{{i_0}^-} = \{M_j \in M^{i_0} : \ n_j^{i_0} < o_j\} \equiv \emptyset, \qquad (17)$$

and outputs the minimum possible flow β_M needed to impose the overall optimum on system M that equals

$$\beta_M = \frac{\sum_{k=1}^{i_0 - 1} r^{k^-}}{r^1} = \frac{r^1 - r^{i_0}}{r^1}, \ i_0 \geq 1.$$

2.3 Modified `OpTop` for arbitrary networks (Algorithm `MOP`)

We consider an arbitrary network G with single-source vertex s and sink t such that the latency function $\ell_e(\cdot)$ per edge e is strictly increasing on flow x_e, differentiable and $x_e \ell_e(x_e)$ is convex (i.e. *standard* latencies). There is a total of flow r to be routed from s to t. The unique optimum routing O of total flow r from vertex s to t can be computed in polynomial time, on any such network G, see in [25] Section 2.3 Fact 2.3.6. This also holds for the Nash assignment N on G, selfishly routed from vertex s to t, see [25] Section 2.5, Remark 2.5.2 (d).

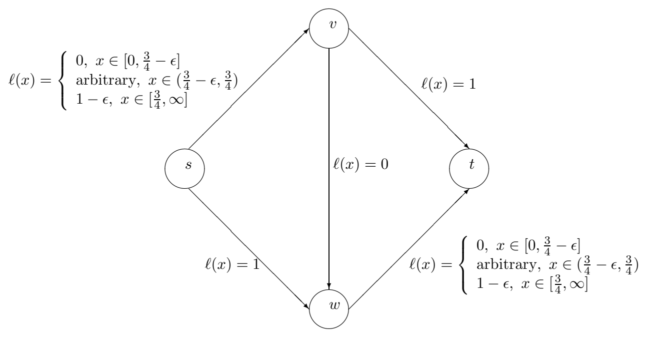
The approach: Consider the optimum assignment O of flow r that wishes to travel from source vertex s to sink t. O assigns flow o_e incurring latency $\ell_e(o_e)$ per edge $e \in G$. Let $\mathcal{P}_{s \to t}$ the set of all $s \to t$ paths. We can compute in polynomial time the shortest paths in $\mathcal{P}_{s \to t}$ with respect to costs $\ell_e(o_e)$ per edge $e \in G$. That is, the paths that given flow assignment O attain latency: $\min_{P \in \mathcal{P}_{s \to t}} \left(\sum_{e \in P} \ell_e(o_e)\right)$ i.e., minimize their latency. It is crucial to observe that, if we want the *induced* Nash assignment by the Stackelberg strategy to attain the optimum cost, then these shortest paths are *the only choice* for selfish users that eager to travel from s to t. Furthermore, the uniqueness of the optimum assignment O determines the minimum part of flow which can be selfishly scheduled on these shortest paths. More precisely, let $A_{s \to t} = \{e \in G : e$ belongs in at least one shortest path from, $s \to t\}$ the set of all "fast" edges in paths from s to t. Observe that the flow $o_{e'}$ assigned by the optimum assignment O on *any* "slow" edge $e' \notin A_{s \to t}$ has incentive to change its path (since it is not a shortest one). Then, for each $e' \notin A_{s \to t}$, a Stackelberg strategy *must* freeze its flow $o_{e'}$ on it. Otherwise, selfish users traversing e' will opt for a shortest path and eventually ruin the overall optimum assignment O. However, for each fast edge $e \in A_{s \to t}$ the flow o_e assigned by O has *no* incentive to change path (it currently is on a shortest one). Therefore, it is useless to employ any Stackelberg strategy on any $e \in A_{s \to t}$. We conclude that the minimum flow sufficient by a Leader to induce the optimum cost equals $\sum_{e' \notin A_{s \to t}} o_{e'}$. It is easy, but tedious, to extend our proof methodology of Sections 2.1 and 2.2 in order to fully justify our argument of correctness as stated above.

```
Algorithm: MOP (Arbitrary s-t Nets)
(1) Initialize Stackelberg strategy
      $S = \langle s_1 = 0, \ldots, s_m = 0 \rangle$ and
      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 $\ell_e(o_e)$, $\forall e \in G$.
      Let $A_{s \to t}$ the set of edges in shortest paths.
(5) For each edge $e_i \notin A_{s \to t}$
      set $s_i = o_i$ and $r_S = r_S + o_i$.
(6) Return the Leader's portion $\beta_G = \frac{r_S}{r}$
      and his strategy $S$.
```

`MOP` achieves coordination ratio 1 on the graph in Fig. 1, used by Roughgarden to argue that $\frac{1}{\alpha} \times O$ guarantee is not possible for general (s,t)-networks, see Appendix B.3 in [25]. We first compute the path $P_0 \in \mathcal{P}$ of minimum latency: $\sum_{e \in P_0} \ell_e(o_e) = \min_{P \in \mathcal{P}} \sum_{e \in P} \ell_e(o_e)$, with respect to the optimum O that on each edge $e \in E$ assigns flow o_e. The optimal flows per edge are:

$$\begin{aligned} o_{s \to v} &= \frac{3}{4} - \epsilon, o_{s \to w} = \frac{1}{4} + \epsilon, o_{v \to w} = \frac{1}{2} - 2\epsilon, \\ o_{v \to t} &= \frac{1}{4} + \epsilon, o_{w \to t} = \frac{3}{4} - \epsilon \end{aligned} \qquad (18)$$

Then $P_0 = s \to v \to w \to t$ (see [25] pp. 143, 5th-3th lines before the end), irrespectively of the flow traversing it. $\forall \ e \in E$ such that $e \notin P_0$ `MOP` assigns flow o_e. Therefore edge $s \to w$ gets flow $o_{s \to w}$ and edge $v \to t$ flow $o_{v \to t}$, according to (18). The selfish routing starts. On starting-node s all flow $r = 1$ will opt to travel trough most wanted path P_0. However, on node s the Leader (according to his strategy dictated by the `MOP`) forces $o_{s \to w}$ of $r = 1$ to travel via $s \to w \notin P_0$ (which is an ugly edge and no one wants it). The remaining selfish $r - o_{s \to w}$ flow will still opt for the most preferred path P_0 and will traverse through edge $s \to v$ (this flow is optimal for $s \to v$). The Leader comes up as soon this flow arrives at node v. He pursues (according to his strategy dictated by `MOP`) another $o_{v \to t}$ flow to traverse the ugly $v \to t \notin P_0$ edge, and finally reach the precious destination t. Now, the remaining $r - o_{s \to w} - o_{v \to t}$ is free to opt for edge $v \to w \in P_0$ which is still highly appealing (this flow is optimal for $v \to w$). A great surprise arises as soon as these $r - o_{s \to w} - o_{v \to t}$ liberians reach node w: they meet their $o_{s \to w}$ fellows. Now the $r - o_{v \to t}$ flow will keep on walking via $w \to t$ (this is optimal for $w \to t$) and reach destination t, where they meet their $o_{v \to t}$ friends. Thus the coordination ratio equals to 1, since all edges get the optimal flows depicted in (18).



$\ell(x) = \begin{cases} 0, \ x \in [0, \frac{3}{4} - \epsilon] \\ \text{arbitrary}, \ x \in (\frac{3}{4} - \epsilon, \frac{3}{4}) \\ 1 - \epsilon, \ x \in [\frac{3}{4}, \infty] \end{cases}$
v
$\ell(x) = 1$
s
$\ell(x) = 0$
t
$\ell(x) = 1$
w
$\ell(x) = \begin{cases} 0, \ x \in [0, \frac{3}{4} - \epsilon] \\ \text{arbitrary}, \ x \in (\frac{3}{4} - \epsilon, \frac{3}{4}) \\ 1 - \epsilon, \ x \in [\frac{3}{4}, \infty] \end{cases}$

Figure 1: A bad example for Stackelberg routing.

3. SINGLE-COMMODITY NETWORKS WITH PARALLEL LINKS, FOCUSING ON HARD INSTANCES $(M, R, \alpha < \beta_M)$

Consider an instance (M, r) with latency functions $\ell_i(x) = a_i x + b_i$, with $a_1 = \ldots, a_m,\ i = 1, \ldots, m$, i.e. parallel linear latency functions, ordered from faster to slower, that is $b_i < b_{i+1}$, $i = 1, \ldots, m-1$. In Lemma 2 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 2. *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 is the induced Nash Assignment. 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. 2. The partial cost of $S + T$ on subsystem $\{M_1, M_2\}$ equals:

$$\begin{aligned} s_1\ell_1(s_1) + (s_2 + t_2)\ell_2(s_2 + t_2) &= \\ s_1\ell^1 + (s_2 + t_2)\ell^2 &= A \\ \text{with } \ell^1 \geq \ell^2 & \end{aligned} \tag{19}$$

Our purpose is to reassign the Leader's flow $s_1 + s_2$ on subsystem $\{M_1, M_2\}$, in a way that $t_1 > 0$, inducing partial cost $\leq A$ and latency $\leq \ell^2$ on the most appealing link. In this way, we get another Leader strategy, with cost $\leq A$, that *satisfies* the property described in Lemma 2. We start by interchanging 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. 3. This decreases the latency on M_1 to $\ell^{1'} < \ell^2$ and increases it on M_2 to $\ell^{2'} > \ell^1$. Since we have parallel linear load functions, we can remove load ϵ 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. 4. The new

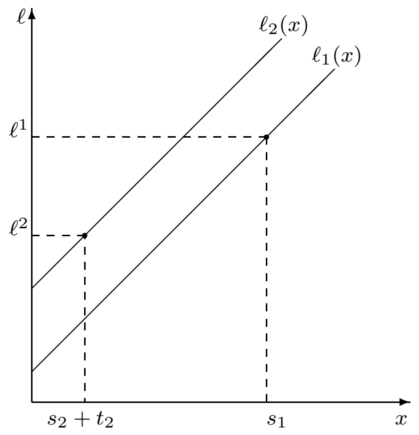


Figure 2: Representation of assignments and latencies on M_1 and M_2 links. The s_1 assignment gives latency ℓ_1 while s_2+t_2 assignment gives latency $\ell_2 \leq \ell_1$.

cost on $\{M_1, M_2\}$ is:

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

□

4. 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.

5. REFERENCES

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

[2] 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.

[3] A. Czumaj. *Selfish routing on the Internet.* In J. Leung, editor, Handbook of Scheduling. CRC Press, Boca Raton, FL, 2004.

[4] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA. ACM/SIAM*, pages 413–420, 2002.

[5] 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.

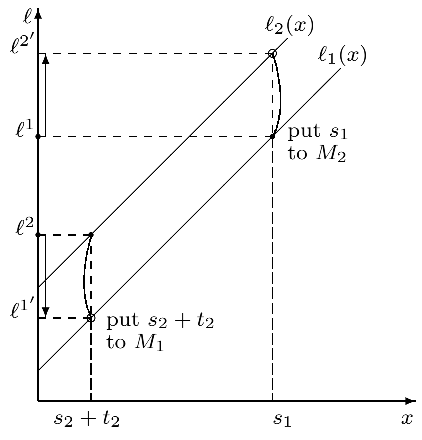


Figure 3: Representation of assignments and latencies on M_1 and M_2 links. Interchange of load in M_1 and M_2, i.e. s_1 goes to M_2 raising the latency to $\ell_2' > \ell_1$ while $s_2 + t_2$ goes to M_1 diminishing the latency to $\ell_1' < \ell_2$.

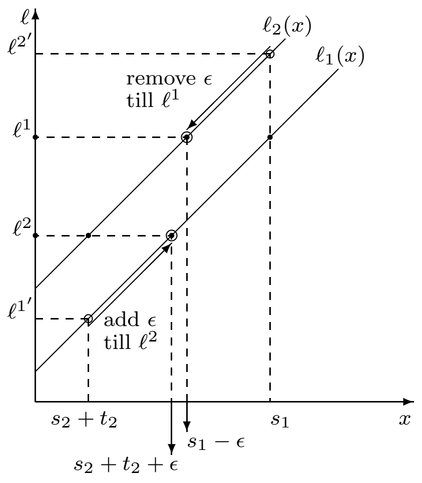


Figure 4: Representation of assignments and latencies on M_1 and M_2 links. Remove from the M_2 link load ϵ till the latency goes to ℓ_1 and add the same load ϵ to M_1 till latency goes to ℓ_2. The cost of this new assignment is less or equal to the cost of the assignment in Fig. 2.

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

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

[8] 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 *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pages 123–134, London, UK, 2002. Springer-Verlag.

[9] Y. Korilis, A. Lazar, and A. Orda. The designer's perspective to noncooperative networks. In *Proceedings of the IEEE INFOCOM '95*, 1995.

[10] 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.

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

[12] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *(STACS '99) Lecture Notes in Computer Science*, 1563:404–413, 1999.

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

[14] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 510–519, New York, NY, USA, 2001. ACM Press.

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

[16] N. Nisan. Algorithms for selfish agents: Mechanism design for distributed computation. In *16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–15, Trier, Germany, 1999. Springer-Verlag. Lecture Notes in Computer Science, LNCS 1563.

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

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

[19] G. Owen. *Game Theory.* Academic Press, Orlando, FL, third edition, 1995.

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

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

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

[23] T. Roughgarden. Stackelberg scheduling strategies. In *33rd Annual Symposium on Theory of Computing*,

pages 104–113. ACM, 2001.

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

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

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

[27] 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.

[28] Slides presentation of “stackelberg scheduling strategies” presented in Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing, 2001. http://theory.stanford.edu/∼tim/slides/`stack_cornell.pdf`.

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