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

A.C. Kaporis, P.G. Spirakis

2006
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 $\alpha r$ 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 [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 $\beta_M$ 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 $\alpha r$ flow can induce $\leq \frac{1}{\beta}$ 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.

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].

Particularly 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 in-
finitesimal 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 \((\tilde{M}, r)\). Then,

\[ C(N) = \epsilon(\tilde{M}, r) \times C(O) \quad (1) \]

where \( \epsilon(\tilde{M}, r) \) depends only on instance \((\tilde{M}, r)\) and can be arbitrarily larger than \([2, 26]\), but if all links in \( M \) have linear load depended latency functions, then \( \epsilon(\tilde{M}, r) \leq 4/3 \). We try to obtain a more clear picture of this degradation on system’s performance, measured by the factor \( \epsilon(\tilde{M}, r) \), 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 \((\tilde{M}, r, \alpha)\), \( 0 \leq \alpha \leq 1 \). This means that each scheduling instance \((\tilde{M}, r)\) corresponds to a family of Stackelberg scheduling instances \((\tilde{M}, r, \alpha)\), parameterized with respect to \( \alpha \in [0, 1] \). Given a Stackelberg scheduling instance \((\tilde{M}, r, \alpha)\), the goal of the Leader is to find an assignment (strategy) \( S \) of his flow on \( M \), such that to induce a Fol lowers’s assignment \( T \) of the remaining \((1-\alpha)r\) flow, with cost \( C(S+T) \) near to the optimum cost \( C(O) \) one. That is

\[ C(S+T) \leq \epsilon(\tilde{M}, r, \alpha) \times C(O) \quad (2) \]

Let us use the name “a posteriori anarchy cost” for the quantity \( \epsilon(\tilde{M}, r, \alpha) \). 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 \( \epsilon(\tilde{M}, r, \alpha) \leq \frac{1}{\alpha} \), \( 0 < \alpha \leq 1 \) for arbitrary latency functions, and if restricted to instances with linear latencies then \( \epsilon(\tilde{M}, r, \alpha) \leq \frac{1}{\alpha} \). From Expression (2), we realize that the portion \( \alpha \) captured by the Leader “pays” an upper bound on system’s degradation factor \( \epsilon(\tilde{M}, r, \alpha) \) 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 \((\tilde{M}, r, \alpha)\), 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 \((\tilde{M}, r, \alpha)\) there may exist a better Leader’s strategy \( S' \) inducing \( T' \) such that \( C(S'+T') \leq C(S+T) \), 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 \( C(S''+T'') \leq C(O) \). 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 \((\tilde{M}, r, \alpha)\) is \( NP \)-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 \( ar \) portion on some edges, inducing a selfish \( s \to t \) routing of the remaining flow with best possible cost? In [25] Appendix B.3 Proposition B.3.1 exhibited a simple 4-node 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 \( \alpha \) 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 \( \beta_M \) 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 \((M, r, \alpha)\) 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 \( \epsilon(\tilde{M}, r, \alpha) \geq \beta_M \) is precisely 1. On the contrary, for all Stackelberg scheduling 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 \( \epsilon(\tilde{M}, r, \alpha < \beta_M) > 1 \), 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 \( \beta_M \) 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 \( \ell(x) = a_i x + b_i \) satisfying the property: \( b_1 \leq \cdots \leq b_m \) and \( a_1 \cdots = a_m \).

(ii) Arbitrary single-commodity nets. Our second main result is an algorithm that efficiently computes the minimum portion \( \beta_G \) 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 \( \text{OpTop} \) 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 \( X = (x_i)_{i=1}^m \) 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 \( X \in \mathbb{R}_+^m \) on \((M, r)\) instance is \( C(X) = \sum_{i=1}^m x_i \ell_i(x_i) \). The minimum cost is incurred by a unique assignment \( O \in \mathbb{R}_+^m \), called the Optimum assignment. The unique assignment \( O \in \mathbb{R}_+^m \) 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 \( k \) source-destination vertex pairs \((s_1, t_1), \ldots, (s_k, t_k)\) and no self loops are allowed. \( \mathcal{P} \) is the set of all paths amongst \((s_i, t_i), i = 1, \ldots, k \), and \( \mathcal{P} = \bigcup \mathcal{P}_i \). A flow is a function \( f : \mathcal{P} \rightarrow \mathbb{R}^+ \). 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 \( \mathcal{P}_i \), \( i = 1, \ldots, k \). 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 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 \( P \in \mathcal{P} \) 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 \( C(f) \) 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 \( i \in \{1, \ldots, k\} \) 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 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 \( \beta_M \) 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.** We give a more efficient algorithm \( \text{OpTop} \) 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 \( O:=\{o_1, \ldots, o_m\} \) the optimum assignment and \( N:=\{n_1, \ldots, n_m\} \) the Nash assignment on \((M, r)\). Intuitively, \( \text{OpTop} \) 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 \( \text{OpTop} \) 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: \( \text{OpTop} \) (Parallel links)**

1. Set \( r_0 = r \) the total flow in \( M \). Compute the optimum assignment \( O := \{o_i : M_i \in M\} \) on instance \((M, r_0)\). Set \( M' \equiv \emptyset \).
2. Compute the Nash assignment \( N := \{n_i : M_i \in M\} \) 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 \( x_i \ell_i(x_i) \) convex on \( x \). Algorithm \( \text{OpTop} \) computes the minimum portion \( \beta_M \) 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 \( N = (n_1, \ldots, n_m) \) with \( \sum_{i=1}^m n_i = r \), and \( O = (o_1, \ldots, o_m) \) with \( \sum_{i=1}^m o_i = r \). We give a more useful definition for the Nash assignment \( N \).

**Definition 1.** An assignment \( N = (n_1, \ldots, n_m) \) 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 \( S = (s_1, \ldots, s_m) \) 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 \( T = (t_1, \ldots, t_m) \) with \( \sum_{i=1}^m t_i = (1 - \beta) r \).

**Definition 2.** Given Stackelberg strategy \( S = (s_1, \ldots, s_m) \) with \( \sum_{i=1}^m s_i = \beta r \) and \( \beta \in [0, 1] \), the assignment \( T = (t_1, \ldots, t_m) \) 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 \( \ell_i(t_i + s_i) \geq L^S \).
The Stackelberg Strategy $S$ induces the Nash assignment $T$. The assignment $T + S$ is called Stackelberg Equilibrium. The Cost of an assignment $X = \{x_1, \ldots, x_m\}$ on $M$ equals $C(X) = \sum_{i=1}^m x_i t_i(x_i)$. The cost of the Stackelberg Equilibrium $S + T$ is $C(S + T) = \sum_{i=1}^m (s_i + t_i) t_i(s_i + t_i)$.

Definition 3. Link $M_i \in M$ is called over-loaded if $n_i > \alpha_i$, under-loaded if $n_i < \alpha_i$, 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 $s_i + t_i = \alpha_i$ for each $i = 1, \ldots, m$. Intuitively, strategy $S$ biases 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 $\exists M_i \in M^{N'^+}$ such that $n'_i > n_i$. Let the Nash assignment $N$ of $(M, r)$.

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

Case 2: If $M_i \in M^{N^+}$ then $\ell_i(n'_i) = L^{N'} > \ell_i(n_i) = \ell_i(0) \geq L^N$. 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.

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 $C(N) \geq 2 C(O)$.

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 $\sum_{i=1}^m s_i = 3r, r \in [0, 1]$, 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 $(1 - 3\beta)r$. Then it holds $n_i = s_i + t_i$ for each $M_i \in M$, 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 $M_i \in M$ that receives load $n_j \geq 0$ 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 $\sum_{i=1}^m s_i = 3r, r \in [0, 1]$. Then, the initial system of links $M$ is transformed by $S$ to the equivalent system $M^S$, such that each link $M_i^S \in M^S$ with load $x$ now experiences latency $\ell_j^S(x) = \ell_j(x + s_j)$. Since for each $M_i \in M$ it holds $s_j \leq n_j$ then $3 \geq r$ such that $t_j = n_j - s_j$ and also $\sum_{i=1}^m t_i = (1 - 3\beta)r$. 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: $\ell_j^S(t_j) = \ell_j(n_i - s_j) + s_j = \ell_j(n_i) = L^N$, for each $M_i \in M$ with $t_j \geq 0$. This means that $T$ is a Nash equilibrium on system $M^S$ and also $S + T \equiv N$.

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 $s_i \geq n_j$ 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 $n_i$, $i \in \{1, \ldots, m\}$.

Theorem 3. 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 $\sum_{i=1}^m s_i = \beta r$ 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 $M$, $L^N > 0$ 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.
\[ \ell^S_j(t_j) = \ell_j(t_j + s_j) = \ell_j(t_j). \]  

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

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

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

\[ \ell^S_j(t_j^{S^-}) = \ell_j(t_j^{S^-}) = L^{S^-} \leq \ell_j(n_j) = L^N. \]  

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

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 \( \geq C(N). \) 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 \( M_j \)).

**Lemma 1.** Let the Nash assignment \( N = (n_1, \ldots, n_m) \) on instance \((M, r). \) Suppose that for a Stackelberg strategy \( S = (s_1, \ldots, s_m) \) 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 \( T = (t_1, \ldots, t_m) \) 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

\[ \ell^S_i(t_i) = \ell_i(t_i + s_i) \geq \ell_i(n_i) \geq L^N. \]  

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

\[ r^S = r - \sum_{M_j \in M^{S^+} \cup M^{S^-}} s_i \leq r^S = r - \sum_{M_i \in M^{S^+}} s_i \leq \sum_{M_i \in M^{S^-}} n_i. \]

where the rightmost inequality stems from the fact that

\[ r^S = \sum_{M_i \in M^{S^+}} s_i \geq \sum_{M_i \in M^{S^+}} n_i. \]

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 \( S \) 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 Nash assignment (that is, by assigning previously \( S \) to subsystem \( M^{S^+} \)) when scheduling the remaining of \( r^S \) appearing in (8) only on to subsystem \( M^{S^-} \). Let \( n^S_i \) (or \( t^S_i \)) 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 \( s_i \leq n^S_i \). 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 \( s_i + t^S_i = n^S_i \). Furthermore, from Inequality (8) we realize that

\[ r^S = \sum_{M_i \in M^{S^-}} n^S_i \leq \sum_{M_i \in M^{S^-}} n_i. \]

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

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

\[ t^S_i = n^S_i - s_i \]  

(see its validity below) otherwise we set \( t^S_i = 0 \). Clearly, each link \( M_i \in M^{S^-} \) with \( t^S_i > 0 \) experiences a common latency.
Given strategy $S$, the Nash assignment $S^o = (s^o_1, \ldots, s^o_n)$ and the Nash assignment of flow $r^0$ onto subsystem $M$ (where $M^o$ is the initial system $M$). Also, let $O = \{o_1, \ldots, o_m\}$ denote the optimal assignment of flow $r$ onto system $M$ (notice that $O$ is not parameterized with respect to the $i$th Phase).

Let the system $M = \{M_1, \ldots, M_m\}$ and the Nash assignment $N = \{n_1, \ldots, n_m\}$ on $(M, r)$. Fix a Stackelberg strategy $S = \{s_1, \ldots, s_n\}$ such that either $s_j \geq n_j$ or $s_j < n_j, j = 1, \ldots, m$. Let $M^+$ be the subset of frozen links $M^+ = \{M_j \in M : s_j \geq n_j\}$, $j = 1, \ldots, m$, and their frozen load $r^+ = \sum_{M_j \in M^+} s_j$. Then the initial Stackelberg game of flow $r$ on $M$ can be simplified to scheduling the remaining unfrozen flow $r^0 = r - r^+$ to the remaining unfrozen subsystem of links $M^o = M \setminus M^o$.

### 2.2 The optimal evolution of $\mathsf{OpTop}$

#### 2.2.1 Phase 1: $\mathsf{OpTop}$ loads optimally all initially under-loaded links.

During Phase $i \geq 1$, let $N^i = \{n^i_1, \ldots, n^i_m\}$ denote the Nash assignment of flow $r^i$ (where $r^i$ equals the initial total flow $r$) on to subsystem of links $M^i$ (where $M^i$ is the initial system $M$). Also, let $O = \{o_1, \ldots, o_m\}$ denote the optimal assignment of flow $r$ onto system $M$ (notice that $O$ is not parameterized with respect to the $i$th Phase).

We introduce the following partition (according to Definition 3) of the system $M^i = \{M_1, \ldots, M_m\}$ during Phase 1

$$M_1^- = \{M_j \in M^i : \; n^i_j < o_j\}$$

$$M_1^+ = \{M_j \in M^i : \; n^i_j \geq o_j\}.$$

According to Theorem 3 and Lemma 1, if during Phase 1 a Stackelberg strategy $S^1 = (s^1_1, \ldots, s^1_m)$ assigns load $s^1_j$ such that $o_j < n^1_j < s^1_j$ 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 $s^1_j > o_j$, irrespectively of any assignment $S^1$ may assign to any other link $M_k \neq M_j$. 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.

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

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

### Simplification of the initial game

Each initially under-loaded link $M_j \in M^1$ will remain frozen to its induced $S^1$ optimum load $s^1_j = o_j$. 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 $\mathsf{OpTop}$ needs a portion $r^+$ to frozen the links in $M^1 \setminus r^1 = \sum_{M_j \in M^1 - o_j}$. 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 $i ≥ 2$: the recursive nature of OptTop. During Phase 2, we consider the Nash assignment $N^2 = \langle n^2_j : M_j \in M^2 \rangle$ when scheduling the remaining flow $r^2 = r^1 - r^{1−}$ of the simplified system $M^2 = M_1 \setminus M^1$ and, similarly as in Phase 1, let

$$M^{2−} = \{ M_j \in M^2 : n^2_j < o_j \}$$

$$M^{2+} = \{ M_j \in M^2 : n^2_j ≥ o_j \}.$$  

Then, applying similarly as in Section 2.2.1 Theorem 2 and Lemma 1 and Proposition 1, OptTop 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, OptTop 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, OptTop terminates as soon as it reaches a Phase 0 where the simplified subsystem $M^0$ has the property

$$M^{0−} = \{ M_j \in M^0 : n^0_j < o_j \} \equiv \emptyset,$$  

(17) and outputs the minimum possible flow $\beta_M$ needed to impose the overall optimum on system $M$ that equals

$$\beta_M = \sum_{j=1}^{k-1} \frac{p_j}{2} = \frac{r^1 - o_0}{r^1}, \quad i_0 ≥ 1.$$  

2.3 Modified OptTop for arbitrary networks (Algorithm MOP) We consider an arbitrary network $G$ with single-source vertex $s$ and sink $t$ such that the latency function $t_e(\cdot)$ per edge $e$ is strictly increasing on flow $x_e$, differentiable and $x, t_e(x)$ 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 $t_e(o_e)$ per edge $e \in G$. Let $P_{s→t}$, the set of all $s→t$ paths. We can compute in polynomial time the shortest paths in $P_{s→t}$, with respect to costs $t_e(o_e)$ per edge $e \in G$. That is, the paths that given flow assignment $O$ attain latency: $\min_{P \in P_{s→t}} \left( \sum_{e \in P} t_e(o_e) \right)$ 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 $A_{s→t} = \{ e \in G : e \text{ belongs in at least one shortest path from } s \text{ to } t \}$ 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 $e′ \notin A_{s→t}$ has incentive to change its path (since it is not a shortest one). Then, for each $e′ \notin A_{s→t}$, 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 $e \in A_{s→t}$, 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 $e \in A_{s→t}$. We conclude that the minimum flow sufficient by a Leader to induce the optimum cost equals $\sum e′ \in A_{s→t} o_e′$. 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 $S = (s_1 = 0, \ldots, s_m = 0)$ and centrally captured flow $r_S = 0$. 

2. Compute the optimum assignment $O := \{ o_e : e \in G \}$ on instance $(G, r)$. 

3. Set cost $t_e(o_e)$ on each edge $e \in G$, $o_e \in O$. 

4. Compute the shortest paths in $G$ with edge-costs $t_e(o_e)$, $\forall e \in G$. 

Let $A_{s→t}$ the set of edges in shortest paths. 

5. For each edge $e \notin A_{s→t}$ set $s_i = o_i$ and $r_S = r_S + o_i$. 

6. Return the Leader’s portion $\beta_G = \frac{r}{r}$ and his strategy $S$. 

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

$$o_{s→v} = \frac{3}{4} - \epsilon, o_{s→w} = \frac{1}{4} + \epsilon, o_{v→w} = \frac{1}{2} - 2\epsilon,$$

$$o_{v→t} = \frac{1}{4} + \epsilon, o_{w→t} = \frac{3}{4} - \epsilon$$  

(18) Then $P_0 = s → v → w → t$ (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 $s → w$ gets flow $o_{s→w}$ and edge $v → t$ flow $o_{v→t}$, according to (18). The selfish routing starts. On starting-node $s$ all flow $r = 1$ will opt to travel through most wanted path $P_0$. However, on node $s$ the Leader (according to his strategy dictated by the MOP) forces $o_{s→w}$ of $r = 1$ to travel via $s → w \notin P_0$ (which is an ugly edge and no one wants it). The remaining selfish $r - o_{s→w}$ flow will still opt for the most preferred path $P_0$ and will traverse through edge $s → v$ (this flow is optimal for $s → v$). TheLeader comes up as soon this flow arrives at node $v$. He pursues (according to his strategy dictated by MOP) another $o_{v→w}$ flow to traverse the ugly $v → t \notin P_0$ edge, and finally reach the precious destination $t$. Now, the remaining $r - o_{s→w} - o_{v→t}$ is free to opt for edge $w → v \notin P_0$ which is still highly appealing (this flow is optimal for $v → w$). A great surprise arises as soon as these $r - o_{s→w} - o_{v→t}$ liberians reach node $w$: they meet their $o_{w→t}$ fellows. Now the $r - o_{v→t}$ flow will keep on walking via $w → t$ (this is optimal for $w → t$) and reach destination $t$, where they meet their $o_{v→t}$ friends. Thus the coordination ratio equals to 1, since all edges get the optimal flows depicted in (18).
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 \(M = \{M_1, \ldots, M_n, \ldots, M_m\}\) around some link \(M_0\) 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:

\[
\begin{align*}
{s_1}{\ell_1(s_1)} + (s_2 + t_2)\ell_2(s_2 + t_2) &= s_1\ell^2 + (s_2 + t_2)\ell^2 = A \\
& \text{with } \ell^2 \geq \ell^1
\end{align*}
\]

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 \(\ell^1 > \ell^2\). Since we have parallel linear load functions, we can remove load \(\epsilon\) from \(M_2\) till the latency it experiences drops from \(\ell^2\) to \(\ell^1\) and place it to \(M_2\) raising its latency from \(\ell^1\) to \(\ell^2\), see Fig. 4. The new cost on \(\{M_1, M_2\}\) is:

\[
\begin{align*}
(s_2 + t_2 + \epsilon)\ell_1(s_2 + t_2 + \epsilon) + (s_1 - \epsilon)\ell_2(s_1 - \epsilon) &= (s_2 + \epsilon + t_2)\ell^2 + (s_1 - \epsilon)\ell^1 = A + \epsilon(\ell^2 - \ell^1) \\
& \leq A,
\end{align*}
\]

since \(\ell^2 \leq \ell^1\)

4. ACKNOWLEDGMENTS

We wish to thank an anonymous referee for urging us to extend BP-Top to general nets. We also wish to thank Ioannis Chatzigiannakis for useful ideas.

5. REFERENCES


![Figure 1: A bad example for Stackelberg routing.](image1)

![Figure 2: Representation of assignments and latencies on \(M_1\) and \(M_2\) links. The \(s_1\) assignment gives latency \(\ell_1\) while \(s_2 + t_2\) assignment gives latency \(\ell_2 \leq \ell_1\).](image2)
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 the $M_2$ link load $\epsilon$ till the latency goes to $\ell_1$ and add the same load $\epsilon$ to $M_1$ till latency goes to $\ell_2$. The cost of this new assignment is less or equal to the cost of the assignment in Fig. 2.


