Efficient Computation of Equilibria in Bottleneck Games via Game Transformation

Thomas L. Werth, Heike Sperber, Sven O. Krumke

University of Kaiserslautern, Department of Mathematics, 67663 Kaiserslautern, Germany

Abstract. We study the efficient computation of Nash and strong equilibria in weighted bottleneck games. In such a game different players interact on a set of resources in the way that every player chooses a subset of the resources as her strategy. The cost of a single resource depends on the total weight of players choosing it and the personal cost every player tries to minimize is the cost of the most expensive resource in her strategy, the bottleneck value.
To derive efficient algorithms for finding Nash equilibria in these games, we generalize a tranformation of a bottleneck game into a special congestion game introduced by Caragiannis et al. [1]. While investigating the transformation we introduce so-called lexicographic games, in which the aim of a player is not only to minimize her bottleneck value but to lexicographically minimize the ordered vector of costs of all resources in her strategy.
For the special case of network bottleneck games, i.e., the set of resources are the edges of a graph and the strategies are paths, we analyse different Greedy type methods and their limitations for extension-parallel and series-parallel graphs.

1 Introduction

A common concept in non-cooperative game theory is a Nash equilibrium [2]. In such a state no user can decrease her personal cost by unilaterally changing to another strategy. This does not imply that users cannot improve by changing their strategies simultaneously. Aumann [3] introduced strong equilibria, which are also stable against deviations of coalitions of users.
Up to now, many different special models have been considered in literature. In this paper we restrict ourselfes to symmetric graph theoretic and combinatorical models explicitly, e.g., network or matroid games, where symmetric means that the sets of possible strategies of all users are the same. In a network game on a directed graph players choose paths from a source to a sink and in a matroid game over some set of resources every player chooses a base of a given matroid.

Congestion games are a very common model considered in literature and were introduced by Rosenthal [4]. In these games, every strategy consists of resources with load dependend latency functions and the personal cost of a player is the sum of the latency values of all resources in her strategy. These games always

have a Nash equilibrium [4].
While in the original work Rosenthal considered a finite number of users, each one with the same positive weight, a game is called weighted if users may have different weights and hence different impact. Then a latency function for a resource depends on the sum of the weights of the users choosing it. For that case Fotakis et al. [6] showed that even symmetric weighted network congestion games do in general not possess a Nash equilibrium, but the restriction to linear latency functions does. Ackermann et al. [7] stated that in a weighted matroid congestion game a Nash equilibrium can be found in polynomial time.

In contrast to congestion games we mainly consider bottleneck games here, where the personal cost of a user is her bottleneck value, i.e., the value of the most expensive resource in her strategy. Network bottleneck games have been introduced in the literature by different authors under different names (e.g., bottleneck routing games by Banner and Orda [8], network load games by Caragiannis et al. [1],games of maximal congestion by Busch and Magdon-Ismail [9]). Banner and Orda [8] established that every weighted network bottleneck game admits a Nash equilibrium. Caragiannis et al. [1] independently proved the existence of Nash equilibria in weighted network bottleneck games with linear latency functions. Harks et al. [10] and Sperber [11] independently generalized these results by showing that every (even weighted) bottleneck game has an optimal strong equilibrium. The optimality here is w.r.t. the network bottleneck, i.e., the maximal congestion on the edges of the network.
Caragiannis et al. [1] also gave a polynomial time algorithm to find a best Nash equilibrium in unweighted network bottleneck games in all graphs with either a single source or a single sink and linear latency functions. Harks et al. [12] showed that strong equilibria can be found in many unweighted bottleneck games in polynomial time by reducing the capacity on the resources with the help of an oracle (e.g., for network games it computes a maximum flow). For weighted bottleneck games it is NP-hard to find a best Nash equilibrium due to a result for weighted congestion games on parallel links by Fotakis et al. [13], since in a graph of parallel links, congestion and bottleneck objectives are the same. Epstein et al. [14] established that on series-parallel graphs all Nash equilibria in the unweighted network bottleneck game are optimal and Sperber [11] showed that all Nash equilibria there are even strong equilibria.

A special case of bottleneck games are lexicographic games. Here, users try to minimize the bottleneck value as well as the second most expensive value and so on. A general procedure to compute optimal solutions was investigated by Della Croce et al. [15].

1.1 Our Contribution

In this work we study atomic symmetric unsplittable bottleneck, lexicographic and congestion games for unweighted and weighted users, respectively. These games may be played on the same instance and differ from each other only by the personal cost of the users.

In order to compute strong equilibria in unweighted bottleneck games we investigate the algorithm of Caragiannis et al. [1] for network bottleneck games with linear latency functions. We generalize it to general bottleneck games with arbitrary non-decreasing functions and call it *bottleneck congestion transformation*. It maps an instance of a bottleneck or lexicographic game to a special instance of a congestion game in such a way that all equilibria in the congestion game with the special instance are also equilibria in the underlying bottleneck or lexicographic game. With the help of this transformation we investigate the connection between bottleneck, lexicographic and congestion games and show how equilibria and optimal solutions in these games are related.

Furthermore, we show that for weighted network bottleneck games in extension-parallel as well as series-parallel graphs an easy GREEDY method computes a Nash equilibrium in polynomial time.

2 Preliminaries and Notation

A weighted game in our context is played by a finite number of k weighted *users* $U = (w_j)_{j=1..k}$ on a finite set of m *resources* E. We will identify users with their weight and assume $w_j = 1$ for all users $j \in U$ for unweighted games. Users cannot choose resources arbitrarily, but they have to follow certain rules: In this symmetric unsplitable game, all users $j \in U$ have the same *set of feasible strategies* $S_j = S \subseteq 2^{|E|}$ and everyone has to choose a single *strategy* $\sigma_j \in S_j$ out of those. Note that we do not allow strategies to contain a resource more than once since it results in equilibria that are not reasonable (see Appendix A for more details). The common *strategy set* is given by $\mathcal{S} = (S_j)_{j=1..k}$.
For every resource $e \in E$ there is a non-decreasing and non-negative *latency function* $\ell_e : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$ that depends on the sum of the weights of the users on this resource, which is also called *load*. When every user $j \in U$ has choosen a strategy $\sigma_j \in S_j$, $\sigma = (\sigma_j)_{j=1..k} \in \mathcal{S}$ is called a *strategy profile*. Then the load on resource $e \in E$ is given by $\sigma_e = \sum_{j=1..k \,:\, e \in \sigma_j} w_j$. Hence, the latency on resource $e \in E$ for the strategy profile σ is given by $\ell_e(\sigma_e)$.

The users in our games have the goal to selfishly minimize their *personal cost* $c_{\sigma_j}(\sigma)$, while some *government* wants to minimize the *social objective* of the game $c(\sigma)$. So, a game is given by the personal costs of the users, a social objective and an instance $\Gamma = [E, U = (w_j)_{j=1..k}, \mathcal{S} = (S_j)_{j=1..k}, (\ell_e)_{e \in E}]$.
It is not clear a priori if there is a situation, where all users and the government can be satisfied. The government is *satisfied*, if the social objective of the strategy profile is minimal and the resulting profile is called an *optimal* strategy.
We call a coalition of users $C \subseteq U$ *satisfied* or *happy* with their strategies $(\sigma_i)_{i \in C}$, if not all of them improve by changing their strategies simultaniously. More formally: when all users i of the coalition C change from their stategies σ_i to other strategy $\tilde{\sigma}_i \in S_i$, resulting in the new strategy profile $(\sigma_{-C}, \tilde{\sigma}_C) \in \mathcal{S}$, then the personal cost of at least one user $j \in C$ does not decrease due to this change, i.e., $c_{\sigma_j}(\sigma) \leq c_{\tilde{\sigma}_j}(\sigma_{-C}, \tilde{\sigma}_C)$.
When all single users, i.e., coalitions of size 1, are happy with their strategies,

the strategy profile is called a *Nash equilibrium*. If even all coalitions of users are happy with their strategies, then the strategy profile is called a *strong equilibrium*. Since single users are coalitions of size one, every strong equilibrium is also a Nash equilibrium.

In a *bottleneck game* the users try to minimize the latency of the most expensive resource in their strategy, i.e., the personal cost user $j \in U$ wants to minimize is defined as *her bottleneck* $b_{\sigma_j}(\sigma) = \max_{e \in \sigma_j} \ell_e(\sigma_e)$. The social objective is choosen to be the *bottleneck of the game*, i.e., the latency of the most expensive resource of all of them: $b(\sigma) = \max_{j \in U} b_{\sigma_j}(\sigma)$.
In a *lexicographic game* users try to minimize not only their bottleneck, but also the second most expensive resource and so on. More formally, let $\sigma_j = (e_1, \ldots, e_\nu)$ be the strategy of user $j \in U$ for some $\nu \in \mathbb{N}$ and let $\pi \in \mathsf{Sym}_\nu$ be a permutation s.t. $\ell_{e_{\pi(i)}}(\sigma_{e_{\pi(i)}}) \geq \ell_{e_{\pi(i+1)}}(\sigma_{e_{\pi(i+1)}})$ for all $i \in \{1, \ldots, \nu-1\}$. Then the personal cost of user j is *her latency vector* $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_j}(\sigma) = \big(\ell_{e_{\pi(1)}}(\sigma_{e_{\pi(1)}}), \ldots, \ell_{e_{\pi(\nu)}}(\sigma_{e_{\pi(\nu)}})\big)$, i.e., the ordered vector of latencies of all of her resources. The social objective is choosen to be the *latency vector of the game* $\boldsymbol{\ell}^{\mathsf{ord}}(\sigma)$, i.e., the ordered vector of latencies of all resources as often as they are used by all users. Let $\#_e(\sigma) = |\{j \in U \mid e \in \sigma_j\}|$ be the number of users choosing resource $e \in E$. Then

$$\boldsymbol{\ell}^{\mathsf{ord}}(\sigma) = \Big(\underbrace{\ell_{e_{\pi(1)}}(\sigma_{e_{\pi(1)}}), \ldots, \ell_{e_{\pi(1)}}(\sigma_{e_{\pi(1)}})}_{\#_{e_{\pi(1)}}(\sigma) \text{ entries}}, \underbrace{\ell_{e_{\pi(2)}}(\sigma_{e_{\pi(2)}}), \ldots}_{\#_{e_{\pi(2)}}(\sigma)}, \ldots, \underbrace{\ell_{e_{\pi(m)}}(\sigma_{e_{\pi(m)}})}_{\#_{e_{\pi(m)}}(\sigma)} \Big),$$

where π is again a permutation s.t. the resources are ordered non-increasingly. Note that the number $\#_e(\sigma)$ of users choosing resource e is only important for our definition of the social objective, whereas the load on the resource, σ_e, is the value determining the latency values and hence the decision of a single user.
In order to minimize the latency vector we have to use some ordering on the whole vector space. The natural choice for the lexicographic game is the *lexicographic order*. Let $x \in \mathbb{R}^\mu_{\geq 0}$ and $y \in \mathbb{R}^\nu_{\geq 0}$ be two vectors and $\eta = \min\{\mu, \nu\}$. Then $x <_{\mathsf{lex}} y$, if there is some $j \in \{1, \ldots, \eta\}$ s.t. $x_j < y_j$ and $x_i = y_i$ for all $i < j$ or if $x_i = y_i$ for all $i \in \{1, \ldots, \eta\}$ and $\mu < \nu$. If the two vectors may be equal, we write $x \leq_{\mathsf{lex}} y$. Analogously we define $x >_{\mathsf{lex}} y$ as $y <_{\mathsf{lex}} x$ and $x \geq_{\mathsf{lex}} y$ as $y \leq_{\mathsf{lex}} x$.
In a *congestion game*, each user tries to minimize the sum of the latencies of the resources in her strategy. So, the personal cost of user $j \in U$ is defined as the *latency of her strategy*: $\ell_{\sigma_j}(\sigma) = \sum_{e \in \sigma_j} \ell_e(\sigma_e)$. For the social objective we choose the *latency of the game*, i.e., the sum of the latencies of all resources used by any user: $\ell(\sigma) = \sum_{j \in U} \ell_{\sigma_j}(\sigma) = \sum_{e \in E} \#_e(\sigma) \ell_e(\sigma_e)$. Note that the personal cost are summed up without a weight factor and thus the latencies on the edges are weighted with the number of users $\#_e(\sigma)$ but not the load σ_e.

2.1 Network Routing Games

A big part of this work deals with network (routing) games in *directed graphs* $G = (V, E)$, where V is the set of n *vertices* and E is the set of m *directed edges*.

For these games we introduce some special notation.
The set of strategies S_j, user $j \in U$ can choose her strategy from, is the set of *s-t-paths* $\mathcal{P}$, where s is a common *source* vertex and t a common *sink* vertex. The resources are the edges of the graph. For some path P, we denote by $P[u, v]$ the subpath of P from vertex u to vertex v. Since we only use this notation when we consider *elementary paths*, i.e., paths that traverse all vertices at most once, the subpath above is uniquely defined.
When all users have choosen their paths, the resulting *flow* f in the graph is denoted by $f = (f_{P_j})_{j=1..k}$, where $f_{P_j} = w_j$. Given such a flow, the *load* or *flow value* on edge $e \in E$ is given by $f_e = \sum_{j=1..k, e \in P_j} f_{P_j}$. For constructive approaches and some proofs it is helpful to have a notation for small changes in the flow f. So, δ_{P_j} denotes one additional unit of load on path P_j and hence $f + w\,\delta_{P_j}$ is a flow sending $w + f_{P_j}$ units of load on path P_j for user j.

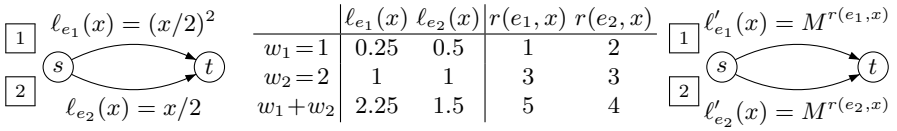
For network games some methods use the structure of the graph. One of the topologies in cosideration are *series-parallel* graphs: A single edge $e = (s, t)$ is defined to be *series-parallel* with source vertex s and sink vertex t. For $i \in \{1, 2\}$, let G_i be series parallel with source vertex s_i and sink vertex t_i.
Then the graph $S(G_1, G_2)$ obtained by identifying t_1 as s_2 is also defined to be *series-parallel* with source vertex s_1 and sink vertex t_2 and is called a *series composition* of G_1 and G_2.
The graph $P(G_1, G_2)$ obtained by identifying s_1 as s_2 and t_1 as t_2 is again defined to be *series-parallel* with source vertex s_1 and sink vertex t_1 and is called a *parallel composition* of G_1 and G_2.
A graph is called *extension-parallel*, if it is series-parallel and every series composition is only an extension composition, i.e., it is given as a composition with a graph consisting of a single edge: $S(G_1, e = (s_2, t_2))$ or $S(e = (s_1, t_1), G_2)$.

3 Bottleneck Congestion Transformation

Caragiannis et al. [1] have shown that an optimal Nash equilibrium in the unweighted network bottleneck game in general graphs with linear latency functions can be computed in polynomial time with the help of a transformation into a congestion game with exponential latency functions on an expanded graph. In this section we generalize their transformation by not restricting ourselfes to linear latency functions or the network game scenario and showing that it computes already optimal strong equilibria. We split the transformation into two parts: first a tranformation of the latency functions and then a second transformation to an expanded setting.

3.1 Induced Congestion Game

We start with a very general transformation that defines for every bottleneck (BG) or lexicographic game (LG) a corresponding *induced congestion game* (iCG) with the same strategies and exponential latency functions s.t. equilibria in iCG are also equilibria in the original game.



(a) Instance BG

	$\ell_{e_1}(x)$	$\ell_{e_2}(x)$	$r(e_1,x)$	$r(e_2,x)$
$w_1=1$	0.25	0.5	1	2
$w_2=2$	1	1	3	3
w_1+w_2	2.25	1.5	5	4

(b) Latency values and exponents

(c) Instance iCG

Fig. 1: Instance of a bottleneck game (a) and the induced congestion game (c), as well as tabular with latency values and exponents for every possible load. Here, we have a user of weight 1 and another one of weight 2. The basis is $M = 2$ and the list of different latency values is $L = (0.25, 0.5, 1, 1.5, 2.25)$.

The idea behind this transformation is the following: If a strategy of few resource has a higher bottleneck value than a strategy of many cheap resources, then the last one should also be cheaper in the induced congestion game. So, we scale the latencies with the help of an exponential function to ensure this.
More formally:
Given a game with instance $\Gamma = [E, U = (w_j)_{j=1..k}, \mathcal{S} = (S_j)_{j\in U}, (\ell_e)_{e\in E}]$, the congestion game with instance $\Gamma_{\mathsf{iCG}} = [E, U, \mathcal{S}, (\ell'_e)_{e\in E}]$ is called the *induced congestion game*. It is given by the same data as the original instance except for the latency functions. For resource $e \in E$ the function is $\ell'_e(x) = M^{r(e,x)}$, where the basis $M = \max\{|\sigma_i| \mid \sigma_i \in S_j \text{ for some } j \in U\} + 1$ is a number larger than the cardinality of every strategy. The exponent $r(e, x)$ is a number s.t. resource e with load $x = \sum_{j\in C} w_j$ for some $C \subseteq U$ has the $r(e, x)$-th cheapest latency value among all possible different latency values from the increasingly ordered list L of the set $\{\ell_e(x) \mid e \in E,\ x = \sum_{j\in C} w_j \text{ for some } C \subseteq U\}$. To understand the notation we give a small example of the transformation for the case of a network game in Fig. 1.

The latency functions for the induced congestion game are exponential functions. So it is not clear whether efficient algorithms for this congestion game exist. But we will show later on that the exponential character can be algorithmically avoided for explicit computations for many interesting models (e.g., network or matroid games).

Because of the exponential character of the latency functions in the induced congestion game we can formulate the following statement.

Lemma 1. *Let σ and $\tilde{\sigma}$ be strategy profiles of a bottleneck, lexicographic and induced congestion game. Then for two strategies $\sigma_o \in \sigma$ and $\tilde{\sigma}_o \in \tilde{\sigma}$ it holds:*

$$\ell'_{\sigma_o}(\sigma) \leq \ell'_{\tilde{\sigma}_o}(\tilde{\sigma}) \Leftrightarrow \boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_o}(\sigma) \leq_{\mathsf{lex}} \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_o}(\tilde{\sigma}) \Rightarrow b_{\sigma_o}(\sigma) \leq b_{\tilde{\sigma}_o}(\tilde{\sigma}).$$

The lemma shows that if a strategy is better for a user in the induced congestion game, then it is also better in the underlying bottleneck game.

Corollary 1 (Equilibria in BG and LG can be computed in iCG). *Let a bottleneck or lexicographic game be given. A strategy profile σ is a (strong) Nash equilibrium in the bottleneck game if it is a (strong) Nash equilibrium in the*

induced congestion game. Furthermore, σ is a (strong) Nash equilibrium in the lexicographic game if and only if it is a (strong) Nash equilibrium in the induced congestion game.

From these statements it follows that (strong) Nash equilibria in the lexicographic game are (strong) Nash equilibria in the bottleneck game. Furthermore, it implies that every unweighted lexicographic game has a Nash equilibrium, since every unweighted congestion game has one [4]. The converse of the first statement in the corollary is in general not true, since the last implication of Lemma 1 is not true in general (see Appendix B).
For the social objectives defined in this work it also holds:

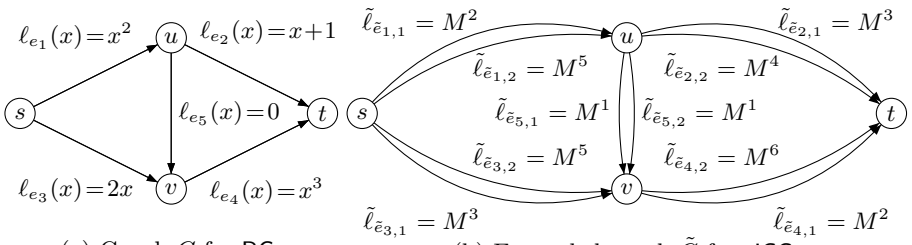
Proposition 1 (Optima in LG are exactly optima in iCG). *A strategy profile σ is optimal in the lexicographic game, if and only if σ is optimal in the induced congestion game.*

We have seen so far that with the help of the transformation we can find equilibria in the bottleneck game as equilibria in the induced congestion game. One can even show that this methods always gives the socially best equilibria, but not necessarily all bad ones. Moreover, some users may be better off in an equilibrium that is worse than another one. For more details about that see Appendix A.1.
The computation of equilibria in congestion games is also not so easy. But, Fabrikant et al. [5] have shown that in network congestion games they can be computed efficiently with the help of a minimum cost flow in an expanded graph. In the next section we will generalize their method.

3.2 Expanded Induced Congestion Game

Here we generalize the second part of the transformation of Caragiannis et al. [1] and state a method s.t. we can find a strong equilibrium in an unweighted bottleneck game as an optimal solution of an *expanded induced congestion game*.

Given an instance of a game $\Gamma = [E, U = \{1, \dots, k\}, \mathcal{S} = (S_j)_{j \in U}, (\ell_e)_{e \in E}]$, the congestion game with instance $\Gamma_{\mathsf{eiCG}} = [\tilde{E}, U, \tilde{\mathcal{S}}, (\tilde{\ell}_e)_{e \in \tilde{E}}]$ is called the *expanded induced congestion game*:
For every resource $e \in E$ we take k copies, called $\tilde{e}_j = [e, j]$ for all $j \in U$, i.e., one for every possible load number. We call all resources $\tilde{e}_j$ parallel to e and denote this by $\tilde{e} \parallel e$. With this the set of resources can be written down explicitly as $\tilde{E} = \{\tilde{e} \mid e \in E, \tilde{e} \parallel e\}$. Then we allocate constant latency values to every resource equal to the latency function in the induced congestion game evaluated at the corresponding load value. More formally, for the j-th copy of resource $e \in E$ the latency value is given by $\tilde{\ell}_{[e,j]} = M^{r(e,j)}$, where the basis is a number larger than the cardinality of every strategy multiplied with the number of users, i.e., $M = k \ \max\{|\sigma_i| \mid \sigma_i \in S_j \text{ for some } j \in U\} + 1$. The exponents $r(e, j)$ are defined as for the induced congestion game.
The set of strategies is the same as in the original instance with the exception that when two users would use the same resource in the original instance then



(a) Graph G for BG (b) Expanded graph $\tilde{G}$ for eiCG

$\ell_{e_i}(x)$	$i=1$	2	3	4	5
$x=1$	1	2	2	1	0
$x=2$	4	3	4	8	0

$r(e_i,x)$	$i=1$	2	3	4	5
$x=1$	2	3	3	2	1
$x=2$	5	4	5	6	1

(c) Latency values and exponents

Fig. 2: Illustration of the expansion process and all notation for the so called Braess graph from (a). Note that paths here have to be resource disjoint to paths of other users. Furthermore, $k=2$, $M=7$ and $L=(0,1,2,3,4,8)$.

they have to pick two different parallel resources in the expanded setting. So the set of common strategies is given by

$$\tilde{\mathcal{S}} = \Big\{\tilde{\sigma}=(\tilde{\sigma}_1,\dots,\tilde{\sigma}_k)\Big|\tilde{\sigma}_j=(\tilde{e}_1,\dots,\tilde{e}_q) \text{ for } j\in U, \sigma_j=(e_1,\dots,e_q)\in S_j, \\ \tilde{E}\ni\tilde{e}_i \parallel e_i\in E, i\in\{1,\dots,q\} \text{ and } |\tilde{\sigma}_{\tilde{e}}|\le 1 \text{ for all } \tilde{e}\in\tilde{E}\Big\}.$$

We call the strategy profile $\tilde{\sigma}$ constructed in the definition *corresponding* to the strategy profile $\sigma=(\sigma_1,\dots,\sigma_k)$. In total, we "take one parallel resource for every user with the latencies from the induced congestion game" and "do not allow users to take common resources". To understand the notation we give an example in Fig. 2.

3.3 Relations of Games

Here we show what the transformation is good for.

Theorem 1 (Optima in eiCG or iCG are optimal SE in BG).

a) If σ is an optimal solution in the induced congestion game, then σ is an optimal strong equilibrium in the bottleneck game. Furthermore, if additionally all latency values are different, i.e., $\ell_e(x)\neq\ell_{e'}(y)$ for all $e\neq e'\in E$ and all $x,y\in\mathbb{N}_0$, then σ is also a strong equilibrium in the lexicographic game and hence a strong equilibrium in the induced congestion game.

b) If $\tilde{\sigma}$ is an optimal solution in the expanded induced congestion game, then the corresponding strategy profile σ is an optimal strong equilibrium in the bottleneck game. Furthermore, if additionally all latency values are different, then σ is a strong equilibrium in the lexicographic game.

We see that in order to find an optimal strong equilibrium in a bottleneck game it is enough to find optimal solutions in special congestion games. Furthermore, one can show that both congestion games we introduced have in general different optimal solutions and equilibria (see Appendix A.2).

3.4 Application

In this section we show that with the help of the expanded bottleneck-congestion tranformation, an optimal strong equilibrium in the unweighted network bottleneck and matroid game can be computed in polynomial time.

Network Routing Game. Given an unweighted network game on a general graph $G = (V, E)$ with general latency functions. The transformation constructs an expanded graph $\tilde{G} = (V, \tilde{E})$ with constant latency values. Then an optimal solution in the congestion game with this instance is computed with a minimum cost flow and decomposed into paths for the single users. This method gives an optimal strong equilibrium in the unweighted network bottleneck game in polynomial time when using a polynomial time minimum cost flow algorithm. In the case of linear latency functions the algorithm works exactly as the one of Caragiannis et al. [1]. Hence, their method does not only compute Nash equilibria, but also strong equilibria.

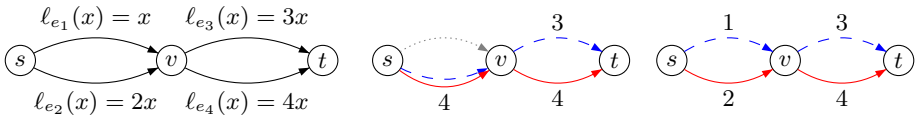
Matroid Game. The same procedure works also in a matroid game (see Ackermann et al. [7]). Here, the resources are resources of a matroid and the strategies of the users are bases of the matroid. A special case here is the network setting, where the resources are edges of an undirected graph and the bases are spanning trees. Roskind and Tarjan [18] have shown that k minimum cost edge disjoint spanning trees can be found in polynomial time and since they do not use any structure of the tree except for the independence property their method can be used for all matroids. This enables us to use the bottleneck congestion transformation to find an optimal strong equilibrium as an optimal solution in the expanded induced matroid congestion game in polynomial time.

4 Greedy Methods

We have already seen in the last section that strong equilibria in unweighted network bottleneck games can be computed in polynomial time with the help of a minimum cost flow algorithm. Now we show that that for special topologies it can be done even faster.

4.1 Extension-Parallel Graphs

In a network of parallel links the bottleneck game and the congestion game coincide. Furthermore, this network is equivalent to scheduling on parallel machines. Here, one just has to choose iteratively the cheapest edges (Greedy strategy) starting with the users with the biggest weight (LPT-rule from scheduling, see



(a) Graph with latencies (b) GREEDY (c) Nash equilibrium

Fig. 3: Shows that the naive GREEDY approach fails in series-parallel graphs. Solid (red) and dashed (blue) lines give the paths of the two users. Numbers are latency values on the edges for the given flow values. In the flow in (b), the dashed user can improve, which results in the flow of (c).

e.g., Koutsoupias et al. [17]).
Since general graphs may consist of many paths from the source to the sink and these may have a different number of edges each, we need some additional notation. We call a path with minimal bottleneck value a shortest bottleneck path or a *narrowest path*. In acyclic graphs and hence especially in extension-parallel graphs, these can be computed directly with the help of a topological sorting in linear time. To do so, one introduces distance labels $d(v_j)$ for all vertices v_j, $j \in \{1, \ldots, k\}$, sorted by a topological sorting, with the recursion:

$$d(v_1) = 0, \quad d(v_j) = \min\left\{\max\left\{d(v_i), \ell_{(v_i,v_j)} \middle| (v_i, v_j) \in E\right\}\right\} \tag{1}$$

for $j \in \{2, \ldots, k\}$. Note that every edge is taken into account only once.

Our algorithm routes the users one after the other in non-ascending order of their weights, i.e., starting with the heaviest ones. It iteratively computes a narrowest path w.r.t. the latency values that result from the load of the users already routed and the current user. We call this method $\text{GREEDY}_{\text{LPT}}$.

Theorem 2 ($\text{Greedy}_{\text{LPT}}$ finds a NE). *In an extension-parallel graph the* $\text{GREEDY}_{\text{LPT}}$ *algorithm finds a Nash equilibrium for the weighted network bottleneck game in time* $\mathcal{O}(k\,m + k \log k)$.

In the unweighted case sorting of the users is not necessary and hence this summand in the running time is canceled. Furthermore, Epstein et al. [14] showed that Nash equilibria in series-parallel graphs are optimal and Sperber [11] showed that they are even strong equilibira. Since extension-parallel graphs are always series-parallel we get the following result:

Corollary 2 (Greedy finds an optimal SE). *In an extension-parallel graph* $\text{GREEDY}_{\text{LPT}}$ *finds an optimal strong equilibrium for the unweighted network bottleneck game in time* $\mathcal{O}(k\,m)$.

4.2 Series-Parallel Graphs

The GREEDY method of the last section does not work in non-extension-parallel graphs as can be seen in the example from Figure 3. The example motivates a modified approach, which takes into account the second, third and so on most

expensive edges, too. So we need *lexicographically shortest paths* (see e.g., [20]). For a s-t-path $P = (e_1, e_2, \ldots, e_d)$ of length $d \in \mathbb{N}$ the *ordered latency vector of this path* is defined to be $\boldsymbol{\ell}_P^{\mathsf{ord}}(f) = \big(\ell_{e_{\pi(1)}}(f_{e_{\pi(1)}}), \ldots, \ell_{e_{\pi(d)}}(f_{e_{\pi(d)}})\big)$, where π is a permutation in the symmetric group Sym_d on d elements s.t. the numbers in the vector are ordered non-increasingly, i.e., $\ell_{e_{\pi(i)}}(f_{e_{\pi(i)}}) \geq \ell_{e_{\pi(i+1)}}(f_{e_{\pi(i+1)}})$ for all $i \in \{1, \ldots, d-1\}$. Path P is called a *lexicographically shortest path* in G, if it has the lexicographic minimal ordered latency vector among all s-t-paths. To compute those, the distance labels are not bottlenecks of the subpaths, see Equation (1), but ordered latency vectors, i.e.,

$$\boldsymbol{d}(v_j) \;=\; \min\left\{\mathsf{sort}\left(\boldsymbol{d}(v_i), \ell_{(v_i,v_j)}\right) \,\middle|\, (v_i, v_j) \in E\right\},$$

where sort is a sorting operator that sorts all the numbers from the vector $\boldsymbol{d}(v_i)$ and the single additional number $\ell_{(v_i,v_j)}$ non-increasingly and outputs a sorted vector, e.g., $\mathsf{sort}\,((4,1),3) = (4,3,1)$. Since the comparison here is a lexicographical one and requires time $\mathcal{O}(n)$ in acyclic graphs, the total complexity for computing a lexicographically shortest path is $\mathcal{O}(m\,n)$.

The algorithm for computing equilibria here is just a little modification of $\textsc{Greedy}_{\mathrm{LPT}}$. Instead of a narrowest path it uses a lexicographically shortest path. Hence, we call it $\textsc{Greedy}_{\mathrm{LPT\text{-}Lex}}$.

Theorem 3 ($\textsc{Greedy}_{\mathrm{LPT\text{-}Lex}}$ finds a NE). *In a series-parallel graph the* $\textsc{Greedy}_{\mathrm{LPT\text{-}Lex}}$ *algorithm finds a Nash equilibrium for the weighted network bottleneck game in time* $\mathcal{O}\left(k(m\ n + \log k)\right)$.

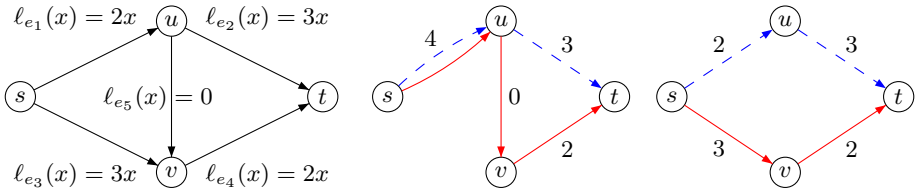
Corollary 3 ($\textsc{Greedy}_{\mathrm{Lex}}$ finds an optimal SE). *In a series - parallel graph* $\textsc{Greedy}_{\mathrm{LPT\text{-}Lex}}$ *finds an optimal strong equilibrium for the unweighted network bottleneck game in time* $\mathcal{O}(k\,m\,n)$.

In this unweighted case with linear latency functions, one can also use the algorithm of Caragiannis et al. [1]. It computes a minimum cost flow in an expanded series-parallel graph (see the section about eiCG), which can be done in time $\mathcal{O}(k\ m\ n + k\ m \log(k\ m))$ by using a Greedy method as Bein et al. [16] showed. However, we do not need the restriction to linear latency functions and the expanded graph and get a small improvement in the running time.

We know from Valdes et al. [19] that every acyclic (directed) graph that is not series-parallel contains a subgraph that is homeomorphic to the Braess graph. The example from Fig. 4 shows that the lexicographic Greedy method does not work on non-series-parallel graphs.

References

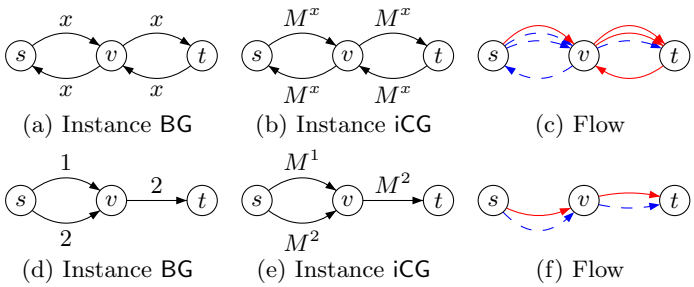
1. I. Caragiannis, C. Galdi, C. Kaklamanis, Network load games, in: ISAAC. LNCS 3827, Springer, 809–818 (2005).
2. J. F. Nash, Equilibrium points in n-person games, Proceedings National Academy of Sciences 36, 48–49 (1950).
3. R. Y. Aumann, Acceptable points in general cooperative n-person games, Contributions to the Theory of Games IV, 287–324 (1959).



(a) Braess Graph (b) $\textsc{Greedy}_{\text{LEX}}$ (c) Nash equilibrium

Fig. 4: Shows that $\textsc{Greedy}_{\text{LEX}}$ fails in a Braess Graph. In (b) and (c) there are two paths (dashed (blue) and solid (red) lines give one path each), one for every unweighted user. Numbers near the edges are latency values for the given flow values. In the flow from (b) the dashed user can improveto the flow from (c).

4. R. W. Rosenthal, A class of games possessing a pure-strategy nash equilibrium, International Journal of Game Theory 2 (1), 65–67 (1973).
5. A. Fabrikant, C. Papadimitriou, K. Talwar, The complexity of pure nash equilibria, in: STOC, 604–612, (2004).
6. D. Fotakis, S. Kontogiannis, P. Spirakis, Symmetry in network congestion games: Pure equilibria and anarchy cost, in: WAOA. LNCS 3879, 161–175, (2005).
7. H. Ackermann, H. Röglin, B. Vöcking, Pure nash equilibria in player-specific and weighted congestion games, Theor. Comput. Sci. 410 (17), 1552–1563 (2009).
8. R. Banner, A. Orda, Bottleneck routing games in communication networks, Journal On Selected Areas In Communications 25 (6), 1173–1179 (2007).
9. C. Busch, M. Magdon-Ismail, Atomic routing games on maximum congestion, Theor. Comput. Sci. 410 (36), 3337–3347 (2009).
10. T. Harks, M. Klimm, R. H. Möhring, Strong nash equilibria in games with the lexicographical improvement property, in: WINE. LNCS 5929, 463–470 (2009).
11. H. Sperber, Complexity of equilibria in games on networks, Part II: Strong equilibria in bottleneck and congestion games, Ph.D. thesis, TU Kaiserslautern (2009).
12. T. Harks, M. Hoefer, M. Klimm, A. Skopalik, Computing pure nash and strong equilibria in bottleneck congestion games, in: ESA. LNCS 6347, 29–38 (2010).
13. D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, P. Spirakis, The structure and complexity of nash equilibria for a selfish routing game, Theor. Comput. Sci. 410 (36), 3305–3326 (2009).
14. A. Epstein, M. Feldman, Y. Mansour, Efficient graph topologies in network routing games, Games and Economic Behavior 66 (1), 115–125 (2009).
15. F. Della Croce, V. T. Paschos, A. Tsoukiàs, An improved general procedure for lexicographic bottleneck problems, Oper. Res. Lett. 24 (4), 187–194 (1999).
16. W. W. Bein, P. Brucker, A. Tamir, Minimum cost flow algorithm for series-parallel networks, Discrete Applied Mathematics 10, 117–124 (1985).
17. E. Koutsoupias, C. H. Papadimitriou, Worst-case equilibria, in: STACS. LNCS 1563, 404–413 (1999).
18. J. Roskind, R. E. Tarjan, A note on finding minimum-cost edge-disjoint spanning trees, Mathematics of Operations Research 10 (4), 701–708 (1985).
19. J. Valdes, R. E. Tarjan, E. L. Lawler, The recognition of series-parallel digraphs, Siam Journal On Computing 11 (2), 298–313 (1982).
20. P. T. Sokkalingam, Y. P. Aneja, Lexicographic bottleneck combinatorial problems, Oper. Res. Lett. 23, 27–33 (1998)



(a) Instance BG (b) Instance iCG (c) Flow

(d) Instance BG (e) Instance iCG (f) Flow

Fig. 5: Two instances (a-c) and (d-f)for two users of weight 1. In th first two columns, the expression is the constant latency function. In the last colum the solid (red), dashed (blue) line give one user each. This flow is a Nash equilibrium in the bottleneck game, but not in the induced congestion game. Here $M = 3$.

Appendix

Here we give additional examples and statements that are mentioned but not shown explicitly in the paper.

A Examples for Bottleneck Congestion Transformation

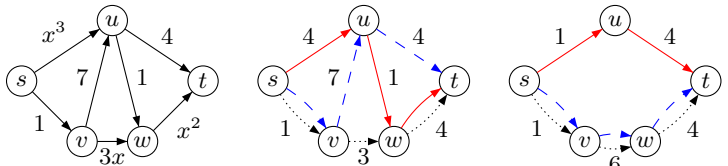
The example of a network game from Fig. 5 (a-c) shows why strategies that use resources more than once are excluded for bottleneck games. The reason is that these would give Nash equilibria that are practically not meaningful. The example also shows that these equilibria cannot always be found with the bottleneck congestion transformation. This also happens without strategies that contain resources more than once as the example from Fig. 5 (d-f) shows.

A.1 Quality of Equilibria

We have seen in the example from Fig. 5 that not all Nash equilibria in our bottleneck games can be found by knowing the appropriate equilibria in the induced congestion games. Luckily, we find the best equilibria with this method:

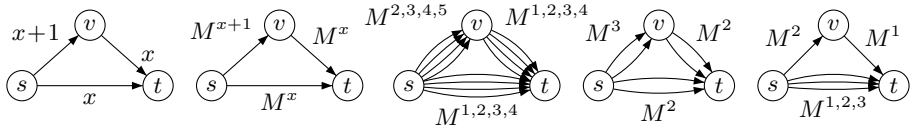
Proposition 2. *Given a bottleneck game and a (strong) Nash equilibrium σ with ordered vector of user bottlenecks given by $\boldsymbol{b}^{\mathsf{ord}}(\sigma) = (b_j)_{j\in U}$. Then there is a (strong) Nash equilibrium $\tilde{\sigma}$ in the induced congestion game with ordered vector of user bottlenecks in the bottleneck game given by $\boldsymbol{b}^{\mathsf{ord}}(\tilde{\sigma}) = (\tilde{b}_j)_{j\in U} \leq_{\mathsf{lex}} \boldsymbol{b}^{\mathsf{ord}}(\sigma)$.*

Proof. If σ is a Nash equilibrium in the lexicographic game, we are done. So assume that it is not. So there are users that can improve by changing their strategy. In every improvement step, the vector of the users latency values decreases lexicographically. After a finite number of changes the resulting strategy profile, called $\tilde{\sigma}$, is a Nash equilibrium in the lexicographic game. So, this $\tilde{\sigma}$ is a Nash equilibrium in the bottleneck game and the vector of all user bottlenecks



(a) Instance BG (b) SE in BG (c) SE in LG

Fig. 6: Modified Braess graph for 3 users of weight 1 with strong equilibria (SE) in the bottleneck game (b and c) and the lexicographic game (only c). In (a), the expression near the edge e is the latency function $\ell_e(x)$. In (b) and (c) there are three paths (dashed (blue), solid (red) and dotted (black)), one for each user and the number is the latency value on the edge for the given load. The dotted user is worse off in (c).



(a) Instance BG (b) Instance iCG (c) Instance eiCG (d) Opt iCG (e) Opt eiCG

Fig. 7: **Opt in eiCG not opt in iCG and vice versa**. Graph for 4 users in the games and optimal solutions. The expressions in (a) and (b) are latency functions or constant latency values for the given load respectively. The load is given by the number of parallel edges; in the expanded setting it is at most 1. In (c) and (e), the upper most edge has the first number as an exponent, the second edge the second number and so on.

is lexicographically not higher than before.
The result for strong equilibria follows along the same lines. □

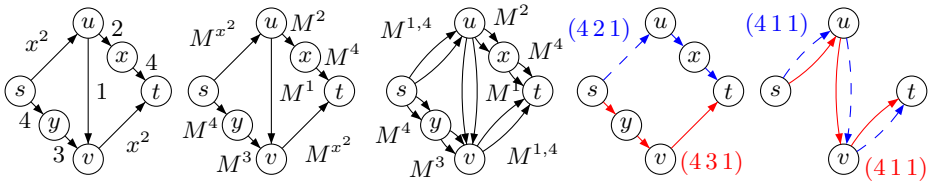
This statement implies that the transformation method always finds the socially best equilibria w.r.t. the lexicographic social objective. But, there may be users that are worse off in the lexicographically better equilibrium (see Fig. 6).

A.2 Relations of Games

Many other implications than those in Theorem 1 are not true. For example, optimal solutions in the expanded induced congestion game are not optimal in the induced congestion game and vice versa (see Fig. 7). Furhermore, strong equilibria in the two congestion games do not coincide in general and might be non-optimal (Fig. 8).

B Proofs for Bottleneck Congestion Transformation

Here we give the proofs for all statements in the section about the bottleneck congestion transformation together with the statements itself.



(a) Instance BG (b) Instance iCG (c) Instance eiCG (d) opt SE iCG (e) opt SE eiCG

Fig. 8: **Opt in eiCG not opt in iCG and vice versa**. Graph for 2 users in the games and strong equilibria. The expressions in (a) and (b) are latency functions or constant latency values for the given load in (c). The numbers in (d) and (e) are the ordered exponents of the latencies of the whole path, where one user has the dashed (blue) line and the other one the solid (red) line.

Lemma 1. *Let σ and $\tilde{\sigma}$ be strategy profiles of a bottleneck, lexicographic and induced congestion game. Then for two strategies $\sigma_o \in \sigma$ and $\tilde{\sigma}_o \in \tilde{\sigma}$ it holds:*

$$\ell'_{\sigma_o}(\sigma) \leq \ell'_{\tilde{\sigma}_o}(\tilde{\sigma}) \Leftrightarrow \boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_o}(\sigma) \leq_{\mathsf{lex}} \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_o}(\tilde{\sigma}) \Rightarrow b_{\sigma_o}(\sigma) \leq b_{\tilde{\sigma}_o}(\tilde{\sigma}).$$

Proof. Let σ and $\tilde{\sigma}$ be strategy profiles of the games and let $\sigma_0 \in \sigma$ and $\tilde{\sigma}_0 \in \tilde{\sigma}$. We start with the equivalence:

$\boxed{\Longrightarrow}$: Let $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) >_{\mathsf{lex}} \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma})$ and assume that $\ell'_{\sigma_0}(\sigma) \leq \ell'_{\tilde{\sigma}_0}(\tilde{\sigma})$, Furthermore, let $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) = \big(\ell_{e_1}(\sigma_{e_1}), \ldots, \ell_{e_\mu}(\sigma_{e_\mu})\big)$ and $\boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma}) = (\ell_{\tilde{e}_1}(\tilde{\sigma}_{\tilde{e}_1}), \ldots, \ell_{\tilde{e}_\nu}(\tilde{\sigma}_{\tilde{e}_\nu}))$ for some $\mu, \nu < M$. Let ρ be choosen in such a way that $\ell_{e_i}(\sigma_{e_i}) = \ell_{\tilde{e}_i}(\tilde{\sigma}_{\tilde{e}_i})$ for all $i \in \{1, \ldots, \rho - 1\}$ and $\ell_{e_\rho}(\sigma_{e_\rho}) > \ell_{\tilde{e}_\rho}(\tilde{\sigma}_{\tilde{e}_\rho})$ or $\rho = \nu + 1$. Hence, for the numbers r in the exponents of the induced latency functions ℓ' it holds $r(e_i, \sigma_{e_i}) = r(\tilde{e}_i, \tilde{\sigma}_{\tilde{e}_i})$ for all $i \in \{1, \ldots, \rho - 1\}$ and $r(e_\rho, \sigma_{e_\rho}) > r(\tilde{e}_\rho, \tilde{\sigma}_{\tilde{e}_\rho})$ or $r(e_\rho, \sigma_{e_\rho}) \geq 0$ for $\rho = \nu + 1$. Because of the ordering of the resources for all $j \in \{\rho + 1, \ldots, \nu\}$ it holds $r(\tilde{e}_j, \tilde{\sigma}_{\tilde{e}_j}) \leq r(\tilde{e}_\rho, \tilde{\sigma}_{\tilde{e}_\rho}) \leq r(e_\rho, \sigma_{e_\rho}) - 1$. So, we can calculate the difference between the latencies of the two strategies:

$$\begin{aligned}
\ell'_{\sigma_0}(\sigma) - \ell'_{\tilde{\sigma}_0}(\tilde{\sigma}) &= \sum_{e \in \sigma_0} \ell'_e(\sigma_e) - \sum_{\tilde{e} \in \tilde{\sigma}_0} \ell'_{\tilde{e}}(\tilde{\sigma}_{\tilde{e}}) && \text{(2a)}\\
&= M^{r(e_\rho, \sigma_{e_\rho})} + \sum_{j=\rho+1}^{\mu} \underbrace{M^{r(e_j, \sigma_{e_j})}}_{\geq 0} - \underbrace{\sum_{j=\rho}^{\nu}}_{< M \text{ summands}} \underbrace{M^{r(\tilde{e}_j, \tilde{\sigma}_{\tilde{e}_j})}}_{\leq M^{r(e_\rho, \sigma_{e_\rho})-1}} && \text{(2b)}\\
&> M^{r(e_\rho, \sigma_{e_\rho})} - M\, M^{r(e_\rho, \sigma_{e_\rho})-1} = 0. && \text{(2c)}
\end{aligned}$$

In (a) we plug in, in (b) the first ρ terms cancel and the others can be split up as discussed above. This gives a contradiction to our assumption.

$\boxed{\Longleftarrow}$: Let $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) \leq_{\mathsf{lex}} \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma})$ and denote the resources again with $\mu, \nu < M$ s.t. $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) = \big(\ell_{e_1}(\sigma_{e_1}), \ldots, \ell_{e_\mu}(\sigma_{e_\mu})\big)$ and $\boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma}) = (\ell_{\tilde{e}_1}(\tilde{\sigma}_{\tilde{e}_1}), \ldots, \ell_{\tilde{e}_\nu}(\tilde{\sigma}_{\tilde{e}_\nu}))$.If the two vectors coincide lexicographically, i.e., $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) = \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma})$, then by the definition of the lexicographic order it holds that the vectors are really equal, i.e.,

$\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) = \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma})$. So $\mu = \nu$ and for all $i \in \{1, \dots, \mu\}$ it holds $\ell_{e_i}(\sigma_{e_i}) = \ell_{\tilde{e}_i}(\tilde{\sigma}_{\tilde{e}_i})$ and hence $r(e_i, \sigma_{e_i}) = r(\tilde{e}_i, \tilde{\sigma}_{\tilde{e}_i})$. So, $\ell'_{\sigma_0}(\sigma) = \ell'_{\tilde{\sigma}_0}(\tilde{\sigma})$.If the two vectors are not equal, i.e., $\boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) <_{\mathsf{lex}} \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma})$, we can use the calculation above with swapped σ and $\tilde{\sigma}$ and get $\ell'_{\sigma_0}(\sigma) < \ell'_{\tilde{\sigma}_0}(\tilde{\sigma})$.

Now we prove the last implication:

$\boxed{\Longrightarrow}$: Since the first element in the ordered vector of the latencies of all resources must be the bottleneck, i.e., there is $e_1 \in \sigma_0$ s.t. $b_{\sigma_0}(\sigma) = \ell_{e_1}(\sigma_{e_1})$ and there is $\tilde{e}_1 \in \tilde{\sigma}_0$ s.t. $b_{\tilde{\sigma}_0}(\tilde{\sigma}) = \ell_{\tilde{e}_1}(\tilde{\sigma}_{\tilde{e}_1})$ it follows immediately that $b_{\sigma_0}(\sigma) \leq b_{\tilde{\sigma}_0}(\tilde{\sigma})$. □

The converse of the last implication is wrong, since $(3,2) >_{\mathsf{lex}} (3,1)$ does not imply $3 > 3$, but the equivalence is also true for equality or strict inequality, i.e.,

$$\ell'_{\sigma_0}(\sigma) \underset{(<)}{=} \ell'_{\tilde{\sigma}_0}(\tilde{\sigma}) \qquad \Longleftrightarrow \qquad \boldsymbol{\ell}^{\mathsf{ord}}_{\sigma_0}(\sigma) \underset{(<_{\mathsf{lex}})}{=} \boldsymbol{\ell}^{\mathsf{ord}}_{\tilde{\sigma}_0}(\tilde{\sigma}).$$

Proposition 1. (Optima in LG are exactly optima in iCG) *A strategy profile σ is optimal in the lexicographic game, if and only if σ is optimal in the induced congestion game.*

Proof. Let $\sigma^* = (\sigma^*_j)_{j=1..k}$ be an optimal solution of the lexicographic game with latency vector $\boldsymbol{\ell}^{\mathsf{ord}}(\sigma^*) = \left(\ell^*_1, \dots, \ell^*_{\mu^*}\right)$. Then for any other strategy profile σ with latency vector $\boldsymbol{\ell}^{\mathsf{ord}}(\sigma) = (\ell_1, \dots, \ell_\mu)$ we have $\boldsymbol{\ell}^{\mathsf{ord}}(\sigma) \geq_{\mathsf{lex}} \boldsymbol{\ell}^{\mathsf{ord}}(\sigma^*)$.
If strict inequality holds then there is ν, the first index where the two vectors differ, i.e., $\ell_i = \ell^*_i$ for all $i < \nu$ and $\ell_\nu > \ell^*_\nu$ or $\mu > \nu = \mu^*$. We name $r = r(e, \sigma_e)$ for some $e \in E$ with $\ell_e(\sigma_e) = \ell_\nu$. With $\#_e = \#_e(\sigma)$ and $\#^*_e = \#_e(\sigma^*)$ we can calculate the difference between the objectives of both strategies:

$$\begin{aligned}
& \ell'(\sigma) \; - \; \ell'(\sigma^*) && \text{(3a)}\\
= & \sum_{\substack{e \in E \\ \ell_e(\sigma_e) > \ell_\nu}} \#_e \; M^{r(e,\sigma_e)} + \sum_{\substack{e \in E \\ \ell_e(\sigma_e) = \ell_\nu}} \#_e \; \underbrace{M^{r(e,\sigma_e)}}_{= M^r} + \underbrace{\sum_{\substack{e \in E \\ \ell_e(\sigma_e) < \ell_\nu}} \#_e \; M^{r(e,\sigma_e)}}_{\geq 0} && \text{(3b)}\\
& - \sum_{\substack{e \in E \\ \ell_e(\sigma^*_e) > \ell_\nu}} \#^*_e \; M^{r(e,\sigma^*_e)} - \underbrace{\sum_{\substack{e \in E \\ \ell_e(\sigma^*_e) = \ell_\nu}} \#^*_e}_{\leq \sum_{\substack{e \in E \\ \ell_e(\sigma_e) = \ell_\nu}} \#_e - 1} \underbrace{M^{r(e,\sigma^*_e)}}_{= M^r} - \underbrace{\sum_{\substack{e \in E \\ \ell_e(\sigma^*_e) < \ell_\nu}} \#^*_e}_{\leq \sum_{e \in E} |\sigma^*_e| \leq \sum_{j=1}^{k} |\sigma^*_j| < M} \underbrace{M^{r(e,\sigma^*_e)}}_{\leq M^{r-1}} && \text{(3c)}\\
> \; & M^r - M\, M^{r-1} \; = \; 0. && \text{(3d)}
\end{aligned}$$

In (b) we split $\ell'(\sigma)$ in three terms and in (c) we split $\ell'(\sigma^*)$ the same way. The first terms of (b) and (c) cancel and the others can be estimated as discussed above. The second term always exists in (b), but not necessary in (c) if there is no edge with cost ℓ_ν there. Then the third term in (c) is needed to compare both sums.

If the two vectors are the same, then the difference above is zero. We see that σ^* is not worse than any strategy profile σ and hence σ^* is optimal in the induced congestion game.

On the other hand, let σ be a non-optimal strategy profile in the lexicographic game. Then there is an optimal strategy σ^*, which is really better than σ. Hence, there is again an index ν for the calculation above. Then σ^* is also better in the induced congestion game and hence σ cannot be optimal there. □

B.1 Main Theorem

Theorem 1. (Optima in eiCG or iCG are optimal SE in BG)

a) If σ is an optimal solution in the induced congestion game, then σ is an optimal strong equilibrium in the bottleneck game. Furthermore, if additionally all latency values are different, i.e., $\ell_e(x) \neq \ell_{e'}(y)$ for all $e \neq e' \in E$ and all $x, y \in \mathbb{N}_0$, then σ is also a strong equilibrium in the lexicographic game and hence a strong equilibrium in the induced congestion game.

b) If $\tilde{\sigma}$ is an optimal solution in the expanded induced congestion game, then the corresponding strategy profile σ is an optimal strong equilibrium in the bottleneck game. Furthermore, if additionally all latency values are different, then σ is a strong equilibrium in the lexicographic game.

Proof. We start with the first part of item *a*). So, let $\sigma = (\sigma_j)_{j \in U}$ be some decomposition of an optimal strategy in the induced congestion game.

Claim: σ is a strong equilibrium in the bottleneck game.
Assume that this is wrong. Then there is some coalition C consisting of users $j \in C$ with their strategies σ_j and bottlenecks $b_j = b_{\sigma_j}(\sigma)$, who can improve by changing to other strategies σ_j^* with new bottlenecks $b_j^* = b_{\sigma_j^*}(\sigma^*) < b_j$. This results in a new strategy $\sigma^* = (\sigma_j^*)_{j \in U} = (\sigma_{-C}, \sigma_C^*)$, where $\sigma_C^* = (\sigma_j^*)_{j \in C}$. W.l.o.g. let the first user have the highest bottleneck value of all the users of the coalition in σ, i.e., $b_1 = \max\left\{ b_j \,\middle|\, j \in C \right\}$. Furthermore, let $e_1 \in \sigma_1$ be a resource on which the highest bottleneck value is attained, i.e., $\ell_{e_1}(\sigma_{e_1}) = b_1$. Let $\mathcal{E}$ be the set of resources used by the users of the coalition in σ, i.e., $\mathcal{E} = \{e \in \sigma_j | \, j \in C\}$. For those resources we have $\sigma_e > 0$. Denote $\#_e = \#_e(\sigma)$ and $\#_e^* = \#_e(\sigma^*)$.

We discuss three types of resources:

i) Resources that are not used by users of the coaltion and have a high latency, i.e., $e \in E \backslash \mathcal{E}$ with $\ell_e(\sigma_e) \geq b_1$. The load of these edges does not change when users of the coalition change their strategies, because no user of the coalition has load on such a resource in σ and no one wants to go to such a resource. So $\#_e^* = \#_e$ and hence $\ell_e(\sigma_e^*) = \ell_e(\sigma_e)$ and $r(e, \sigma_e^*) = r(e, \sigma_e)$ for these resources.

ii) The most expensive bottleneck resources used by users of the coalition, i.e., $e \in \mathcal{E}$ with $\ell_e(\sigma_e) = b_1$. All of these will be left by at least one member of the coalition, because these users have to improve their bottleneck values when they change their strategies. Hence the load and latency of these resources

will decrease, i.e., $\sigma_e^* \leq \sigma_e - 1$, $\ell_e(\sigma_e^*) < \ell_e(\sigma_e)$ and so for the exponent it holds: $r(e, \sigma_e^*) < r(e, \sigma_e) = r$.

iii) For all other resources, i.e., $e \in \mathcal{E}$ with $\ell_e(\sigma_e) < b_1$, the load may change due to the change of the coalition, but the latency does not increase to b_1 and hence, it is really smaller than b_1. So the exponents are smaller than r, i.e., $r(e, \sigma_e^*) < r$.

Note that the numbers $r(e, x)$ are fixed when the resources, latencies and users are given and do not depend on any strategy choices.

With this we can compare the cost of the new strategy σ^* with the old strategy σ in the induced congestion game.

$$\ell'(\sigma^*) = \sum_{\substack{e \in E \setminus \mathcal{E} \\ \ell_e(\sigma_e) \geq b_1}} \#_e^* \underbrace{\ell'_e(\sigma_e^*)}_{=\ell'_e(\sigma_e)} + \sum_{\substack{e \in \mathcal{E} \\ \ell_e(\sigma_e) = b_1}} \#_e^* \underbrace{M^{r(e,\sigma_e^*)}}_{\leq M^{r-1}} + \sum_{\substack{e \in E \\ \ell_e(\sigma_e) < b_1}} \#_e^* \underbrace{M^{r(e,\sigma_e^*)}}_{\leq M^{r-1}} \tag{4a}$$

$$\leq \underbrace{\sum_{\substack{e \in E \setminus \mathcal{E} \\ \ell_e(\sigma_e) \geq b_1}} \#_e \, \ell'_e(\sigma_e) + \sum_{\substack{e \in \mathcal{E} \\ \ell_e(\sigma_e) = b_1}} \#_e \, \ell'_e(\sigma_e) + \sum_{\substack{e \in E \\ \ell_e(\sigma_e) < b_1}} \#_e \, \ell'_e(\sigma_e)}_{=\ell'(\sigma)} \tag{4b}$$

$$- \underbrace{\sum_{\substack{e \in \mathcal{E} \\ \ell_e(\sigma_e) = b_1}} \#_e \, \ell'_e(\sigma_e)}_{\geq M^r, \text{ since } e_1 \in \mathcal{E} \text{ with } \ell_{e_1}(\sigma_{e_1}) = b_1} - \underbrace{\sum_{\substack{e \in E \\ \ell_e(\sigma_e) < b_1}} \#_e \, \ell'_e(\sigma_e)}_{\geq 0} + \underbrace{\sum_{e \in E} \#_e^*}_{=\sum_{j=1}^{k} |\sigma_j^*| < M} M^{r-1} \tag{4c}$$

$$< \ell'(\sigma) \; - \; M^r \; + \; M \, M^{r-1} \; = \; \ell'(\sigma). \tag{4d}$$

In Equation (a) we split in the three cases discussed above. The first term in (b) in the same as the first one in (a). Furthermore, we add the last two terms and substrackt them again in line (c). The last summand in (c) is not less than the two last terms in (a). So, σ^* would have less cost than a minimum cost strategy, which is a contradiction to our assumption. Hence, σ is a strong equilibrium in the bottleneck game.

Claim: σ is also optimal in the bottleneck game.
Assume, this is wrong. Then there must be some strategy σ^* with a smaller bottleneck value, i.e., $b(\sigma^*) < b(\sigma)$. So there are some resources $e_0^* \in \sigma^*$ and $e_0 \in \sigma$ on which the bottleneck values are attained, i.e., $\ell_{e_0^*}(\sigma_{e_0^*}^*) = b(\sigma^*) < b(\sigma) = \ell_{e_0}(\sigma_{e_0})$. Hence, the maximal exponent of the new strategy is really smaller than the one of the old strategy, i.e., $r(e_0^*, \sigma_{e_0^*}^*) < r(e_0, \sigma_{e_0}) = r$. Then

$$\ell'(\sigma^*) = \sum_{e \in E} \#_e^* \underbrace{M^{r(e,\sigma_e^*)}}_{\leq M^{r-1}} \leq \underbrace{\sum_{j=1}^{k} \sum_{e \in \sigma_j^*}}_{< M \text{ summands}} M^{r-1} < \; M^r \; \leq \sum_{e \in E} \#_e \, M^{r(e,\,i)} \; = \; \ell'(\sigma),$$

which again is a contradiction to the fact that σ is a minimum cost strategy in the induced congestion game. Hence, σ is an optimum in the bottleneck game.

Now we show that an optimal solution of the induced congestion game is also a strong equilibrium in the lexicographic game if all latency values are different. The argumentation is up to small changes very similar to the one for the bottleneck game. Let $\sigma = (\sigma_j)_{j \in U}$ be some decomposition of an optimal strategy in the induced congestion game.

Claim: σ is a strong equilibrium in the lexicographic game.
Assume that this is wrong. Then there is some coalition C of users $j \in C$, their strategies σ_j and latency vectors $\boldsymbol{\ell}_j^{\mathsf{ord}} = \boldsymbol{\ell}_{\sigma_j}^{\mathsf{ord}}(\sigma) = \left(\ell_1^{(j)}, \ldots, \ell_{\mu_j}^{(j)}\right)$ of length $\mu_j \in \mathbb{N}$, who can improve by changing to other strategies σ_j^* with new latency vectors $\boldsymbol{\ell}_j^{\mathsf{ord}^*} = \boldsymbol{\ell}_{\sigma_j^*}^{\mathsf{ord}}(\sigma^*) = \left(\ell_1^{(j)^*}, \ldots, \ell_{\mu_j^*}^{(j)^*}\right) <_{\mathsf{lex}} \boldsymbol{\ell}_j^{\mathsf{ord}}$ of length $\mu_j^* \in \mathbb{N}$. This results in a new strategy $\sigma^* = (\sigma_j^*)_{j \in U} = (\sigma_{-C}, \sigma_C^*)$, where $\sigma_C^* = (\sigma_j^*)_{j \in C}$. W.l.o.g. let ℓ_1 be the highest latency value that decreases due to the change of the coalition and ℓ_2^* the hightest latency value to which some latency decreased due to the change of the coalition. Since every resource has different latency values, i.e., $\ell_e(x) \neq \ell_{e'}(y)$ for all $e \neq e' \in E$ and all $x, y \in \mathbb{N}_0$, the most expensive resource some user left, must be really more expensive than the most expensive value she and all other users joined, i.e., $\ell_1 > \ell_2^*$. This is the only point, where we need this assumption and the statement of the theorem is wrong without some statement that ensures that the overall latency vector really drops. Now, it follows for all $e \in E$ with $\ell_e(\sigma_e^*) > \ell_2^*$ that $\ell_e(\sigma_e^*) = \ell_e(\sigma_e)$. Furthermore, let e_1 be a resource with high latency in the beginning, i.e., $\ell_{e_1}(\sigma_{e_1}) = \ell_1$ and e_2^* be a resource with high latency in the new strategy, i.e., $\ell_{e_2^*}(\sigma_{e_2^*}^*) = \ell_2^*$. Since $\ell_{e_1}(\sigma_{e_1}) > \ell_{e_2^*}(\sigma_{e_2^*}^*)$, it holds again that $r = r(e_1, \sigma_{e_1}) > r(e_2^*, \sigma_{e_2^*}^*) = r^*$.

The rest of the argumentation and of the calculation are nearly the same as in the bottleneck case, when we replace the high bottleneck value b_1 by the high latency value ℓ_1. In the case ii), not all of the high values have to decrease, but then we do not have to add and subtrackt them in the calculation. With this, one can show that σ^* would have less cost than a minimum cost strategy in the induced congestion game, which is a contradiction to our assumption. Hence, σ is a strong equilibrium in the lexicographic game.

Next we show part *b*). This is very similar to item *a*), but for the expanded setting. Let $\sigma = (\sigma_j)_{j \in U}$ be some decomposition of the strategy σ corresponding to an optimal strategy $\tilde{\sigma}$ in the expanded induced congestion game. Note that in this expanded game the latencies $\tilde{\ell}_{\tilde{e}}$ of all resources $\tilde{e} \in \tilde{E}$ of the expanded setting are constant and do not depend on the load.

Claim: σ is a strong equilibrium in the bottleneck game.
The argumentation here is exactly the same as in part *a*). We assume that the strategy σ is not a strong equilibrium in the bottleneck game. Then there is some coalition C consisting of users $j \in C$ with their strategies σ_j and bottlenecks $b_j = b_{\sigma_j}(\sigma)$, who can improve by changing to other strategies σ_j^* with new bottlenecks $b_j^* = b_{\sigma_j^*}(\sigma^*) < b_j$. This results in a new strategy $\sigma^* = (\sigma_j^*)_{j \in U} = (\sigma_{-C}, \sigma_C^*)$, where $\sigma_C^* = (\sigma_j^*)_{j \in C}$. W.l.o.g. let the first user have the highest bottleneck value of all of the users of the coalition in the beginning, i.e., $b_1 = \max\left\{b_j \middle| j \in C\right\}$. Furthermore, let $e_1 \in \sigma_1$ be a resource on which the first bottleneck value is

attained, i.e., $\ell_{e_1}(\sigma_{e_1}) = b_1$. Since the order is kept by the latencies and their exponents in the expanded induced congestion game, $r = r(e_1, \sigma_{e_1})$ the highest exponent and $\tilde{\ell}_{\tilde{e}_1}$ is the highest latency of all users of the coalition in the beginning in the expanded setting. Here, $\tilde{e}_1 \parallel e_1$ is one of the most expensive of the parallel resources corresponding to e_1. Let $\mathcal{E}$ be the set of resources used by the users of the coalition in the beginning, i.e., $\mathcal{E} = \{e \in \sigma_j \,|\, j \in C\}$. For those resources we have $\sigma_e > 0$ and hence $\tilde{\sigma}_{[e,i]} = 1$ for all $i \in \{1, \ldots, \sigma_e\}$, i.e., the first σ_e parallel resources $[e, i]$ in eiCG corresponding to e in the expanded game, while for all resources $\tilde{e} \in \tilde{E}$ it holds $\tilde{\sigma}_{\tilde{e}} \in \{0, 1\}$.

We have the same three types of resources as in part a), but the calculation is a little bit different, because we have to calculate in the expanded setting:

$$\tilde{\ell}(\tilde{\sigma}^*) = \sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in E\setminus\mathcal{E} \\ \ell_e(\sigma_e)\geq b_1}} \tilde{\sigma}^*_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}} + \sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in\mathcal{E} \\ \ell_e(\sigma_e)=b_1}} \tilde{\sigma}^*_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}} + \sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in E \\ \ell_e(\sigma_e)<b_1}} \tilde{\sigma}^*_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}} \tag{5a}$$

$$\leq \sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in E\setminus\mathcal{E} \\ \ell_e(\sigma_e)\geq b_1}} \tilde{\sigma}_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}} + \sum_{\substack{e\in\mathcal{E} \\ \ell_e(\sigma_e)=b_1}} \sum_{i=1}^{\sigma_e-1} M^{r(e,i)} + \sum_{\substack{e\in E \\ \ell_e(\sigma_e)<b_1}} \sum_{\substack{i=1 \\ r(e,i)\leq r-1}}^{\sigma^*_e} M^{r(e,i)} \tag{5b}$$

$$\leq \underbrace{\sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in E\setminus\mathcal{E} \\ \ell_e(\sigma_e)\geq b_1}} \tilde{\sigma}_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}} + \sum_{\substack{e\in\mathcal{E} \\ \ell_e(\sigma_e)=b_1}} \sum_{i=1}^{\sigma_e} M^{r(e,i)} + \sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in E \\ \ell_e(\sigma_e)<b_1}} \tilde{\sigma}_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}}}_{=\tilde{\ell}(\tilde{\sigma})} \tag{5c}$$

$$- \underbrace{\sum_{\substack{e\in\mathcal{E} \\ \ell_e(\sigma_e)=b_1}} M^{r(e,\sigma_e)}}_{\geq M^r} - \underbrace{\sum_{\substack{\tilde{e}\in\tilde{E} \\ \tilde{e}\parallel e\in E \\ \ell_e(\sigma_e)<b_1}} \tilde{\sigma}_{\tilde{e}}\ \tilde{\ell}_{\tilde{e}}}_{\geq 0} + \underbrace{\sum_{\substack{e\in E \\ \ell_e(\sigma_e)<b_1}} \sum_{\substack{i=1 \\ r(e,i)\leq r-1}}^{\sigma^*_e} \underbrace{M^{r(e,i)}}_{\leq M^{r-1}}}_{\leq \sum_{j=1}^{k} |\sigma^*_j| < M}$$

$$< \tilde{\ell}(\tilde{\sigma}) - M^r + M^r = \tilde{\ell}(\tilde{\sigma}). \tag{5d}$$

In Equation (a) we split in the three cases discussed above and in (b) we include the relations between σ and σ^* from the three cases. The first and last term in (b) are the same as the first and last one in (c). The second term from (b) is split in the second and fourth one from (c). Furthermore, we add and substrackt the third (and fifth) term in (c). So, $\tilde{\sigma}^*$ would have less cost than a minimum cost strategy of the expanded induced congestion game, which is a contradiction to our assumption. Hence, σ must be a strong equilibrium in the bottleneck game.

Claim: σ is also optimal in the bottleneck game.
Assume, this is wrong. Then there must be some strategy σ^* with a smaller bottleneck value, i.e., $b(\sigma^*) < b(\sigma)$. So there are some resources $e^*_0 \in \sigma^*$ and $e_0 \in \sigma$ on which the bottleneck values are attained, i.e., $\ell_{e^*_0}(\sigma^*_{e^*_0}) = b(\sigma^*) <$

$b(\sigma) = \ell_{e_0}(\sigma_{e_0})$. Hence, the maximal exponent of the new strategy is really smaller than the one of the old strategy, i.e., $r(e_0^*, \sigma_{e_0^*}^*) < r(e_0, \sigma_{e_0}) = r$. Then

$$\tilde{\ell}(\tilde{\sigma}^*) = \sum_{e \in E} \sum_{i=1}^{\sigma_e^*} \underbrace{M^{r(e,i)}}_{\leq M^{r-1}} \leq \underbrace{\sum_{j=1}^{k} \sum_{e \in \sigma_j^*} M^{r-1}}_{< M \text{ summands}} < M^r \leq \sum_{e \in E} \tilde{\sigma}_{\tilde{e}} \sum_{i=1}^{\sigma_e} M^{r(e,i)} = \tilde{\ell}(\tilde{\sigma}),$$

which again is a contradiction to the fact that σ is a minimum cost strategy and hence σ is an optimum.

Claim: σ is a strong equilibrium in the lexicographic game.
We can do the same argumentation and the same computation as in part a). Of cource, we have to replace b_1 by ℓ_1 again. Then we receive a contradiction to $\tilde{\sigma}$ being a minimum cost strategy. Hence, σ is a strong equilibrium in the lexicographic game. □

B.2 Application

Given an unweighted network game on a general graph $G = (V, E)$ with general latency functions. The instance is transformed to an expanded graph $\tilde{G} = (V, \tilde{E})$ with constant latency values. Then a minimum cost flow is computed and decomposed into paths for the single users.

Theorem 4. *The algorithm described above computes an optimal strong equilibrium in the unweighted network bottleneck game. This is done in polynomial time when using a polynomial time minimum cost flow algorithm.*

Proof. Correctness follows directly from Theorem 1.

Running time and recources: The construction of $\tilde{G} = (V, \tilde{E})$ with $\left|\tilde{E}\right| = k\,m$ can be done in polynomial time $\mathcal{O}(k\,m)$ and also requires only polynomial space, although, the latency functions are all exponential. This is because they all have the same basis M and an integral exponent. So, comparing a path P consisting of $l \leq m$ edges in a flow f in G with a single edge e' in a flow f' that is a little bit more expensive than the most expensive edge of the path, it holds $r(e, f_e) \leq r(e', f'_{e'}) - 1$ and hence we have

$$\sum_{e \in P} M^{r(e,f_e)} \leq m\, M^{r(e',f'_{e'})-1} < M^{r(e',f'_{e'})}.$$

Because of the calculation above there is no need to use the full information about the new latency function. Whenever one path has a lexicographic larger vector of non-increasingly ordered latency values, it is more expensive. For this comparison, we only need the exponents and if the highest exponents are equal, the number how often they appear. So, the amount of information needed for comparing is in $\mathcal{O}(m)$. With this, every operation in a minimum cost flow algorithm can be done in polynomial time. Computing the flow in the normal graph G and path decompositions can also be done in polynomial time. □

Algorithm 1: GREEDY$_{\mathrm{LPT}}$

Input : Extension-parallel graph $G = (V, E)$, k users $U = (w_j)_{j=1..k}$, latency functions $(\ell_e)_{e \in E}$
Output : Nash equilibrium f

sort users by their weight s.t. $w_j \geq w_{j+1}$ for all $j \in \{1, \ldots, k-1\}$;
set $f = \emptyset$ and $\bar{\ell}_e = \ell_e(w_1)$;
for $q = 1, \ldots, k$ **do**
    find a narrowest path P in the graph G w.r.t. the latency values $(\bar{\ell}_e)_{e \in E}$;
    set $f = f + w_q \, \delta_P$ and $\bar{\ell}_e = \ell_e(f_e + w_{q+1})$ for all $e \in E$;

Theorem 5. *Given an unweighted matroid game. Then the transformation to* eiCG *and the algorithm from Roskind and Tarjan [18] compute an optimal strong equilibrium in polynomial time.*

Proof. Follows along the very same lines as for the network game. The only point where the latency functions are needed in the algorithm of Roskind and Tarjan [18] is the sorting of the resources in the beginning. Since the basis of the functions is always M, it is enough to compare the exponents and this can be done in linear time. So, in fact, we do not even need the exponential transformation, because the order of the latencies is not affected by the transformation. □

C Proofs for Greedy Methods

Here we give the proofs for all statements about the GREEDY methods.

Theorem 2. (Greedy$_{\mathrm{LPT}}$ finds a NE) *In an extension-parallel graph the* GREEDY$_{\mathrm{LPT}}$ *algorithm finds a Nash equilibrium for the weighted network bottleneck game in time* $\mathcal{O}(k\,m + k \log k)$.

Proof. Observe that the latency values $\bar{\ell}$ used by GREEDY$_{\mathrm{LPT}}$ are the same as the real latency functions ℓ for the given load in the network.
Correctness is shown by induction on the number of users k:
$k = 1$: GREEDY chooses a narrowest path w.r.t. the latency that results from the weight of the first user, so she cannot improve.
$k \to k+1$: Let f be a Nash flow in G for k users and let P be a narrowest path for a new user $k+1$ with weight w and bottleneck value $b = b_P(f + w\,\delta_P)$, all found by GREEDY$_{\mathrm{LPT}}$. Now, assume that $f + w\,\delta_P$ is not a Nash flow.
Then there is a user $j \in \{1, \ldots, k\}$ with weight $w_j \geq w$ on her path P_j, who can improve her bottleneck value $b_j = b_{P_j}(f + w\,\delta_P)$ by unilateraly changing to another path $\tilde{P}_j$ with bottleneck value $\tilde{b}_j = b_{\tilde{P}_j}(f + w\,\delta_P - w_j\,\delta_{P_j} + w_j\,\delta_{\tilde{P}_j}) < b_j$.
Since user j was satisfied before the new user was added to path P, all of her edges with latency equal to the bottleneck value (bottleneck edges) $B_j \subseteq E$ must also be in P and hence $b \geq b_j$.

We show now that user $k+1$ would also have been able to choose path $\tilde{P}_j$. Since she has no higher weight than user j, her bottleneck value will not be

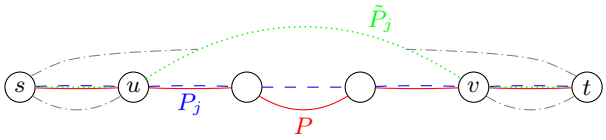


Fig. 9: Illustration of notation and structure of the graph. Solid (red) path P, dashed (blue) path P_j and dotted (green) path $\tilde{P}_j$. Dashed-dotted, gray lines are "forbidden" in extension-parallel graphs.

higher than the one of user j on this new path and hence choosing this path would be also an improvement, which is a contradiction.

Paths in extension parallel graphs may differ at most once since otherwise there is a subgraph homeomorphic to the easiest series-parallel graph that is not extension-parallel (see Milchtaich[1]). So, let u be the unique vertex, where path $\tilde{P}_j$ leaves path P_j and v be the unique vertex, where path $\tilde{P}_j$ enters path P_j. Since all old bottleneck edges B_j are in $P_j \cap P$ and none of them is in the new path $\tilde{P}_j$, $\tilde{P}_j$ and P_j have to differ between u and v. Hence, there cannot be any path from s to u or from v to t that does not coincide with P_j between these vertices. This holds especially for path P and is the crucial point that is not true in general graphs. So, all three paths coincide between s and u and between v and t, i.e., $P[s,u] = \tilde{P}_j[s,u] = P_j[s,u]$ and $P[v,t] = \tilde{P}_j[v,t] = P_j[v,t]$ and differ between u and v, i.e., $P[u,v] \cap \tilde{P}_j[u,v] = \emptyset$ and $P_j[u,v] \cap \tilde{P}_j[u,v] = \emptyset$ (see Figure 9).

Hence the number of users on $\tilde{P}_j$ is the same for the flow routing the new user to path $\tilde{P}_j$ instead of P while all other users stay, resulting in the flow $f + w\,\delta_{\tilde{P}_j}$, and the flow rerouting user j from path P_j to path $\tilde{P}_j$ while all other users keep their strategy, resulting in the flow $f + w\,\delta_P - w_j\,\delta_{P_j} + w_j\,\delta_{\tilde{P}_j}$. Since the weight of the new user is not higher than the weight of user j, i.e., $w \leq w_j$, the load on path $\tilde{P}_j$ is not higher in the first flow and hence the bottleneck is not higher there. So, with the results above, i.e., $\tilde{b}_j < b_j \leq b$, we have in total:

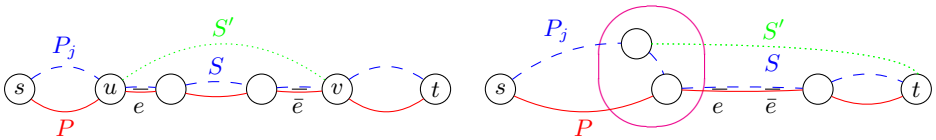
$$b_{\tilde{P}_j}(f + w\,\delta_{\tilde{P}_j}) \;\leq\; b_{\tilde{P}_j}(f + w\,\delta_P - w_j\,\delta_{P_j} + w_j\,\delta_{\tilde{P}_j}) \;=\; \tilde{b}_j \;<\; b_j \;\leq\; b$$

and hence the new user could have choosen path $\tilde{P}_j$ in the beginning. This is a contradiction to the fact that P is a shortest bottleneck path in the algorithm.

The important steps for the running time are sorting of the users, time $\mathcal{O}(k \log k)$ and computing of k narrowest paths, each one in time $\mathcal{O}(m+n)$. □

Theorem 3. (Greedy$_{\text{LPT-Lex}}$ finds a NE) *In a series-parallel graph the* GREEDY$_{\text{LPT-LEX}}$ *algorithm finds a Nash equilibrium for the weighted network bottleneck game in time* $\mathcal{O}(k(m\ n + \log k))$.

[1] I. Milchtaich, Network topology and the efficiency of equilibrium, Games and Economic Behavior 57 (2), pp. 321346 (2006)



(a) Situation in the proof (b) Excluded cases (non-series-parallel)

Fig. 10: Illustration of notation and structure of the graph. Solid (red) path P and dashed (blue) path P_j with dashed (blue) subpath S containing the expensive edges e, $\bar{e}$ and the prefered dotted (green) route S'.

Proof. The latency values $\bar{\ell}$ used in the algorithm give rise to exactly the same values as the latency functions ℓ from G.
Correctness is shown by induction on the number of users k:
$k = 1$: GREEDY$_{\text{LPT-LEX}}$ chooses a lexicographic shortest path, so the single user cannot improve.
$k \to k+1$: Let f be a Nash flow in G for k users and let P be a lexicographically shortest path for user $(k+1)$ with weight w and bottleneck value $b = b_P(f + w\,\delta_P)$, all found by GREEDY$_{\text{LEX}}$. Assume that $f + w\,\delta_P$ is no Nash flow. Then there is a user $j \in \{1, \ldots, k-1\}$ with weight $w_j \geq w$ on her path P_j, who can improve her bottleneck value $b_j = b_P(f + w\,\delta_P)$ by changing some path segment S of her path to another subpath S'. Since user j was satisfied before the new user was added to path P, all bottleneck edges must also be in P and so $b \geq b_j$. Let e be the first and $\bar{e}$ be the last bottleneck edge on P_j and hence on $S \cap P \cap P_j$.

The new path segment S' leaves path P_j before edge e in a vertex u and enter P_j after edge $\bar{e}$ in a vertex v that also P traverses, since otherwise G would not be series-parallel (see Figure 10). Thus, the notation $S = P_j[u, v]$ is justified and $P_j[u, v] \cap S' = \emptyset$, $P[u, v] \cap S' = \emptyset$.
Since user j changes from S to S', we know

$$b_j \;=\; b_S(f + w\,\delta_P) \;>\; b_{S'}(f + w\,\delta_P + w_j\,\delta_{S'} - w_j\,\delta_S).$$

The $(k+1)$-th user cannot be better off than user j, i.e., $b \geq b_j$, and has weight $w \leq w_j$. Changing her flow from $P[u, v]$ to S' would result in a bottleneck on this subpath of

$$b_{S'}\left(f + w\,\delta_P + w\,\delta_{S'} - w\,\delta_{P[u,v]}\right) \;\leq\; b_{S'}\left(f + w\,\delta_P + w_j\,\delta_{S'} - w_j\,\delta_S\right) \;<\; b.$$

So the $(k+1)$-th user did not choose the lexicographically shortest path, which is a contradiction to our assumption.

The important steps for the running time are sorting of the users, time $\mathcal{O}(k \log k)$ and computing of k lexicographic shortes paths, each one in time $\mathcal{O}(m\,n)$. □