# Computing Pure Nash and Strong Equilibria in Bottleneck Congestion Games

Tobias Harks* Martin Hoefer† Max Klimm‡ Alexander Skopalik§

May 14, 2010

**Abstract**

Bottleneck congestion games properly model the properties of many real-world network routing applications. They are known to possess strong equilibria – a strengthening of Nash equilibrium to resilience against coalitional deviations. In this paper, we study the computational complexity of pure Nash and strong equilibria in these games. We provide a generic centralized algorithm to compute strong equilibria, which has polynomial running time for many interesting classes of games such as, e.g., matroid or single-commodity bottleneck congestion games. In addition, we examine the more demanding goal to reach equilibria in polynomial time using natural improvement dynamics. Using unilateral improvement dynamics in matroid games pure Nash equilibria can be reached efficiently. In contrast, computing even a single coalitional improvement move in matroid and single-commodity games is strongly NP-hard. In addition, we establish a variety of hardness results and lower bounds regarding the duration of unilateral and coalitional improvement dynamics. They continue to hold even for convergence to approximate equilibria.

## 1 Introduction

One of the central challenges in algorithmic game theory is to characterize the computational complexity of equilibria. Results in this direction yield important indicators if game-theoretic solution concepts are plausible outcomes of competitive environments in practice. Probably the most prominent stability concept in (non-cooperative) game theory is the Nash equilibrium – a state, from which no player wants to unilaterally deviate – and its' complexity has been under increased scrutiny for quite some time. A drawback of Nash equilibrium is that in general it exists only in mixed strategies. There are, however, practically important classes of games that allow pure Nash equilibria (PNE), most prominently congestion games. In a congestion game [19], there is a set of resources, and the pure strategies of players are subsets of this set. Each resource has a delay function depending on the *load*, i.e., the number of players that select strategies containing the respective resource. The individual cost for a

*Department of Mathematics, TU Berlin, Germany, harks@math.tu-berlin.de.

†Department of Computer Science, RWTH Aachen University, Germany, mhoefer@cs.rwth-aachen.de.
Supported by DFG through UMIC Research Center at RWTH Aachen University and grant Ho 3831/3-1.

‡Department of Mathematics, TU Berlin, Germany. klimm@math.tu-berlin.de.

§Department of Computer Science, RWTH Aachen University, Germany. skopalik@cs.rwth-aachen.de.
Supported in part by the German Israeli Foundation (GIF) under contract 877/05.

player in a regular congestion game is given by the *sum* over the delays of the resources in his strategy.

Congestion games are an elegant model to study the effects of resource usage and congestion with strategic agents. They have been used frequently to model competitive network routing scenarios [20]. For these games the complexity of exact and approximate PNE is now well-understood. A detailed characterization in terms of, e.g., the structure of strategy spaces [1, 8] or the delay functions [4, 22] has been derived. However, regular congestion games have shortcomings, especially as models for the prominent application of routing in computer networks. The throughput of a stream of packets is usually determined by the delay experienced due to available bandwidth or capacity of links. Here the throughput of a player is closely related to the performance of the most congested (bottleneck) link (see, e.g., [3, 5, 13, 18]). A model that captures this aspect more realistically are *bottleneck congestion games*, in which the individual cost of a player is the maximum (instead of sum) of the delays in his strategy. Despite being a more realistic model for network routing, they have not received similar attention in the literature. For classes of non-atomic (with infinitesimally small players) and atomic splittable games (finite number of players with arbitrarily splittable demand) existence of PNE and bounds on the price of anarchy were considered in [5, 16]. For atomic games with unsplittable demand PNE do always exist [3]. In fact, Harks et al. [11] establish the finite improvement property via a lexicographic potential function. Interestingly, they are able to extend these conditions to hold even if *coalitions* of players are allowed to change their strategy in a coordinated way. This implies that bottleneck congestion games do admit even (pure) *strong equilibria (SE)*, a solution concept introduced by Aumann [2]. In a SE, no coalition (of any size) can deviate and strictly decrease the individual cost of each member. Every SE is a PNE, but the converse holds only in special cases (e.g., for singleton games [12]).

SE represent a very robust and appealing stability concept. In general games, however, they are quite rare, which makes the existence guarantee in bottleneck congestion games even more remarkable. For instance, even in dominant strategy games such as the Prisoner's Dilemma there might be no SE. Not surprisingly, for regular congestion games with linear aggregation the existence of SE is not guaranteed [12, 14]. The existence of PNE and SE in bottleneck congestion games raises a variety of important questions regarding their computational complexity. In which cases can PNE and SE be computed efficiently? As the games have the finite improvement property, another important issue is the duration of natural (coalitional) improvement dynamics. More fundamentally, it is not obvious that even a single such coalitional improving move can be found efficiently. These are the main questions that we address in this paper.

### 1.1 Our Results

We examine the computational complexity of PNE and SE in bottleneck congestion games. In Section 2 we focus on computing PNE and SE using (centralized) algorithms. Our first main result is a generic algorithm that computes a SE for any bottleneck congestion game. The algorithm iteratively decreases capacities on the resources and relies on a *strategy packing oracle*. The oracle decides if a given set of capacities allows to pack a collection of feasible strategies for all players and outputs a feasible packing if one exists. The running time of the

algorithm is essentially determined by the running time of this oracle. We show that there are polynomial time oracles for matroids, $a$-arborescences, and single-commodity networks. Thus, our generic algorithm yields an *efficient algorithm* to compute SE for the corresponding classes of games. For general games, however, we show that the problem of computing a SE is NP-hard, even in two-commodity networks.

In Section 3 we study the duration and complexity of sequential improvement dynamics that converge to PNE and SE. We first observe that for every matroid bottleneck congestion game the lazy best response dynamics presented in [1] converge in polynomial time to a PNE. In contrast to this positive result for unilateral dynamics, we show that it is NP-hard to decide if a coalitional improving move exists, even for matroid and single-commodity network games, and even if the deviating coalition is fixed a priori. This highlights an interesting contrast for these two classes of games: While there are polynomial time algorithms to compute a SE, it is impossible to decide efficiently if a given state is a SE – the decision problem is co-NP-hard.

For more general games, we observe in Section 3.2 that constructions of [22] regarding the hardness of computing PNE in regular games can be adjusted to yield similar results for bottleneck games. In particular, in (a) symmetric games with arbitrary delay functions and (b) asymmetric games with bounded-jump delay functions computing a PNE is PLS-complete. In addition, we show that in both cases there exist games and starting states, from which every sequence of improvement moves to a PNE is exponentially long. We extend this result to the case when moves of coalitions of size $\mathcal{O}(n^{1-\epsilon})$ are allowed, for any constant $\epsilon > 0$. In addition, we observe that all of these hardness results generalize to the computation of $\alpha$-approximate PNE and SE, for any polynomially bounded factor $\alpha$. An $\alpha$-approximate PNE (SE) is a relaxation of a PNE (SE), which is stable only against (coalitional) improving moves that decrease the delay of the (every) moving player by at least a factor of $\alpha > 1$.

We conclude the paper in Section 4 by outlining some interesting open problems regarding the convergence to approximate equilibria. All proofs missing in this extended abstract are presented in the Appendix.

### 1.2 Preliminaries

*Bottleneck congestion games* are strategic games $G = (N, \mathcal{S}, (c_i)_{i \in N})$, where $N = \{1, \ldots, n\}$ is the non-empty and finite set of players, $\mathcal{S} = \bigtimes_{i \in N} \mathcal{S}_i$ is the non-empty set of *states* or *strategy profiles*, and $c_i : \mathcal{S} \to \mathbb{N}$ is the *individual cost* function that specifies the cost value of player $i$ for each state $S \in \mathcal{S}$. A game is called finite if $\mathcal{S}$ is finite. For the sake of a clean mathematical definition, we define strategies and costs using the general notion of a congestion model. A tuple $\mathcal{M} = (N, R, \mathcal{S}, (d_r)_{r \in R})$ is called a *congestion model* if $N = \{1, \ldots, n\}$ is a non-empty, finite set of players, $R = \{1, \ldots, m\}$ is a non-empty, finite set of *resources*, and $\mathcal{S} = \bigtimes_{i \in N} \mathcal{S}_i$ is the set of states or profiles. For each player $i \in N$, the set $\mathcal{S}_i$ is a non-empty, finite set of *pure strategies* $S_i \subseteq R$. Given a state $S$, we define $\ell_r(S) = |\{i \in N : r \in S_i\}|$ as the number of players using $r$ in $S$. Every resource $r \in R$ has a *delay function* $d_r : S \to \mathbb{N}$ defined as $d_r(S) = d_r(\ell_r(S))$. In this paper, all delay functions are non-negative and non-decreasing. A congestion model $\mathcal{M}$ is called *matroid congestion model* if for every $i \in N$ there is a matroid $M_i = (R, \mathcal{I}_i)$ such that $\mathcal{S}_i$ equals the set of bases of $M_i$. We denote by $\mathrm{rk}(\mathcal{M}) = \max_{i \in N} \mathrm{rk}(M_i)$ the rank of the matroid congestion model. (Bottleneck) congestion games corresponding to matroid congestion models will be

called *matroid (bottleneck) congestion games*. Matroids exhibit numerous nice properties, some of which are described in the Appendix. For a comprehensive overview see standard textbooks [15, Chapter 13] and [21, Chapters 39 – 42].

Let $\mathcal{M}$ be a congestion model. The corresponding *bottleneck congestion game* is the strategic game $G(\mathcal{M}) = (N, \mathcal{S}, (c_i)_{i\in N})$ in which $c_i$ is given by $c_i(S) = \max_{r\in S_i} d_r\big(\ell_r(S)\big)$. We drop $\mathcal{M}$ whenever it is clear from context. We define the corresponding *regular congestion game* in the same way, the only difference is that $c_i(S) = \sum_{r\in S_i} d_r\big(\ell_r(S)\big)$. For a coalition $C \subseteq N$ we denote by $-C$ its complement and by $\mathcal{S}_C = \bigtimes_{i\in C} \mathcal{S}_i$ the set of states of players in $C$. A pair $\big(S, (S'_C, S_{-C})\big) \in \mathcal{S} \times \mathcal{S}$ is called an *$\alpha$-improving move* of coalition $C$ if $c_i(S) > \alpha c_i(S'_C, S_{-C})$ for all $i \in C$ and $\alpha \geq 1$. For $\alpha = 1$ we call $\big(S, (S'_C, S_{-C})\big)$ *improving move* (or *profitable deviation*). A state $S$ is a $k$-strong equilibrium ($k$-SE), if there is no improving move $(S, \cdot)$ for a coalition of size at most $k$. We say $S$ is a strong equilibrium (SE) if and only if it is a $n$-SE. Similarly, $S$ is a pure Nash equilibrium (PNE) if and only if it is a 1-SE. We call a state $S$ an $\alpha$-approximate SE (PNE) if no coalition (single player) has an $\alpha$-improving move $(S, \cdot)$. We denote by $I(S)$ the set of all possible $\alpha$-improving moves $(S, S')$ to other states $S' \in \mathcal{S}$. We call a sequence of states $(S^0, S^1, \dots)$ an *improvement path* if every tuple $(S^k, S^{k+1}) \in I(S^k)$ for all $k = 0, 1, 2, \dots$. Intuitively, an improvement path is a path in a so-called *state graph* $\mathcal{G}(G)$ derived from $G$, where every state $S \in \mathcal{S}$ corresponds to a node in $\mathcal{G}(G)$ and there is a directed edge $(S, S')$ if and only if $(S, S') \in I(S)$.

# 2 Computing Strong Equilibria

In this section, we investigate the complexity of computing a SE in bottleneck congestion games. We first present a generic algorithm that computes a SE for an arbitrary bottleneck congestion game. It uses an oracle that solves a strategy packing problem (see Definition 2.1), which we term *strategy packing oracle*. For games in which the strategy packing oracle can be implemented in polynomial time, we obtain a polynomial algorithm computing a SE. We then examine games for which this is the case. In general, however, we prove that computing a SE is NP-hard, even for two-commodity bottleneck congestion games.

## 2.1 The Dual Greedy

The general approach of our algorithm is to introduce upper bounds $u_r$ (capacities) on each resource $r$. The idea is to iteratively reduce upper bounds of costly resources as long as the residual capacities admit a feasible strategy packing, see Definition 2.1 below. Our algorithm can be interpreted as a dual greedy, or *worst out algorithm* as studied, e.g., in the field of network optimization, see Schrijver [21].

**Definition 2.1** (Strategy packing oracle)**.**
INPUT: *Finite set of resources $R$ with upper bounds $(u_r)_{r\in R}$, and $n$ collections $\mathcal{S}_1, \dots, \mathcal{S}_n \subseteq 2^R$ given implicitly by a certain combinatorial property.*
OUTPUT: *Sets $S_1 \in \mathcal{S}_1, \dots, S_n \in \mathcal{S}_n$ such that $|i \in \{1, \dots, n\} : r \in S_i| \leq u_r$ for all $r \in R$, or the information, that no such sets exist.*

More specifically, when the algorithm starts, no strategy has been assigned to any player and each resource can be used by $n$ players, thus, $u_r = n$. If $r$ is used by $n$ players, its cost

**Algorithm 1**: Dual Greedy, the strategy packing oracle is denoted by $\mathfrak{O}$.

**Input**: Bottleneck congestion game $G(\mathcal{M})$ to the model $\mathcal{M} = (N, R, \mathcal{S}, d)$
**Output**: SE of $G$

1 **set** $N' = N$, **and** $u_r = n$, $l_r = 0$, $d'_r = d_r$ for all $r \in R$;
2 **while** $\{r \in R : u_r > 0\} \neq \emptyset$ **do**
3  **choose** $r' \in \arg\max_{r \in R: u_r > 0} \{d_r(u_r + l_r)\}$;
4  $u_{r'} := u_{r'} - 1$;
5  **if** $\mathfrak{O}(N', R, \mathcal{S}_{N'}, d', u_r) = \emptyset$ **then**
6  $u_{r'} := u_{r'} + 1$;
7  **foreach** $j \in N'$ **with** $r' \in S'_j$ **do**
8  $S_j := S'_j$;
9  **set** $l_r := l_r + 1$, $u_r := u_r - 1$ for all $r \in S'_j$;
10  $N' := N' \setminus \{j\}$;
11  **end**
12  **foreach** $r \in R$ **do**
13  $d'_r(x) = d_r(x + l_r)$ for all $x = 1, \ldots, n - l_r$;
14  **end**
15  **end**
16  $S' = \mathfrak{O}(N', R, \mathcal{S}_{N'}, d', u_r)$;
17 **end**
18 **return** $S$;

equals $d_r(n)$. The algorithm now iteratively reduces the maximum resource cost by picking a resource $r'$ with maximum delay $d_r(u_r)$ and $u_r > 0$. The number of players allowed on $r'$ is reduced by one and the strategy packing oracle checks, if there is a feasible strategy profile obeying the capacity constraints. If the strategy packing oracle outputs such a feasible state $S$, the algorithm reiterates by choosing a (possibly different) resource that has currently maximum delay. If the strategy packing oracle returns $\emptyset$ after the capacity of some $r' \in R$ was reduced to $u_{r'} - 1$, we fix the strategies of those $u_{r'}$ many players that used $r'$ in the state the strategy packing oracle computed in the previous iteration and decrease the bounds $u_r$ of all resources used in the strategies accordingly. This ensures that $r'$ is *frozen*, i.e., there is no residual capacity on $r'$ for allocating this resource in future iterations of the algorithm. The algorithm terminates after at most $n \cdot m$ calls of the oracle. For a formal description of the algorithm see Algorithm 1.

**Theorem 2.2.** *Dual Greedy computes a SE.*

*Proof.* Let $S$ denote the output of the algorithm. In addition, we denote by $N_k$, $k = 1 \ldots, K$, the sets of players whose strategies are determined after the strategy packing oracle (denoted by $\mathfrak{O}$) returned $\emptyset$ for the $k$-th time. Clearly, $c_i(S) \leq c_j(S)$ for all $i \in N_k$, $j \in N_l$, with $k \geq l$. We will show by complete induction over $k$ that the players in $N_1 \cup \cdots \cup N_k$ will not participate in any improving move of any coalition.

We start with the case $k = 1$. Let $(u_r)_{r \in R}$ be the vector of capacities in the algorithm after the strategy packing oracle returned $\emptyset$ in line 5 for the first time and $u_{r'}$ is updated in line 6.

Suppose there is a coalition $C \subseteq N$ with $C \cap N_1 \neq \emptyset$ that deviates profitably from $S$ to $T = (S'_C, S_{-C})$. We distinguish two cases.

**Case 1**: $\ell_r(T) \leq u_r$ for all $r \in R$. Since $\mathfrak{O}(N, R, \mathcal{S}, d, \tilde{u}) = \emptyset$, where $\tilde{u}_r = u_r - 1$, if $r = r'$ and $u_r$, else, at least $|N_1|$ players use $r'$ in $T$. Using $d_{r'}(T) \geq d_r(S)$ for all $r \in R$, we obtain a contradiction to the fact that every member of $C$ must strictly improve.

**Case 2**: There is $\tilde{r} \in R$ such that $\ell_{\tilde{r}}(T) > u_r$. Using that Dual Greedy iteratively reduces the capacity of those resources with maximum delay (line 3), we derive that $d_{\tilde{r}}(T) \geq d_r(S)$ for all $r \in R$. Using $\ell_{\tilde{r}}(T) > u_r$, there is at least one player $i \in C$ with $\tilde{r} \in T_i$, hence, this player does not strictly improve.

For the induction step $k \to k+1$, suppose the players in $N_1 \cup \cdots \cup N_k$ stick to their strategies and consider the players in $N_{k+1}$. As the strategies of the players in $N_1 \cup \cdots \cup N_k$ are fixed, the same arguments as above imply that no subset of $N_{k+1}$ will participate in a profitable deviation from $S$. □

It is worth noting that the dual greedy algorithm applies to arbitrary strategy spaces. If the strategy packing problem can be solved in polynomial time, this algorithm computes a SE in polynomial time. Hence, the problem of computing a SE is polynomial time reducible to the strategy packing problem. For general bottleneck congestion games the converse is also true.

**Theorem 2.3.** *The strategy packing problem is polynomial time reducible to the problem of computing a SE in a bottleneck congestion game.*

*Proof.* Given an instance of the set packing problem $\Pi$ we construct a bottleneck congestion game $G_\Pi$. Let $\Pi$ be given as set of resources $R$ with upper bounds $(u_r)_{r \in R}$, and $n$ collections $\mathcal{S}_1, \ldots, \mathcal{S}_n \subseteq 2^R$. The game $G_\Pi$ consists of the resource $R \cup \{r_1, \ldots, r_n\}$ and the players $1, \ldots, n+1$. The set of strategies of player $i \in \{1, \ldots, n\}$ is $\{S_i \cup \{r_i\} \mid S_i \in \mathcal{S}_i\}$. Player $n+1$ has the strategies $R$ and $\{r_1, \ldots, r_n\}$. For each resource $r \in R$ The delay is 0 it is used by at most $u_r + 1$ and 2 otherwise. For each resource $r \in \{r_1, \ldots, r_n\}$ the delay is 0, if used by at most one player and 1, otherwise.

If a strategy profile of the player $1, \ldots, n$ violates an upper bound $u_r$ on a resource $r \in R$, player $n+1$ has delay of 2 if he plays strategy $R$. If he plays $\{r_1, \ldots, r_n\}$ he and all other players have delay of 1. Hence, if there is a feasible strategy packing, every SE of the game yields delay 0 for every player. Otherwise, every SE yields delay 1 for every player. Therefore, the state of the players $1, \ldots, n$ in a SE of $G_\Pi$ corresponds to a solution for the strategy packing problem $\Pi$, if such a solution exists. On the other hand, if there is no solution for $\Pi$, every player in every SE in $G_\Pi$ has delay of 1. □

In the next section we will present some interesting cases, in which the strategy packing problem can be solved in polynomial time, or in which computation becomes NP-hard.

## 2.2 Complexity of Strategy Packing

**Theorem 2.4.** *The strategy packing problem can be solved in polynomial time for matroid bottleneck congestion games where the strategy set of player $i$ equals the set of bases of a matroid $M_i = (R, \mathcal{I}_i)$ given by a polynomial independence oracle.*

*Proof.* For each matroid $M_i = (R, \mathcal{I}_i)$, we construct a matroid $M_i' = (R', \mathcal{I}_i')$ as follows. For each resource $r \in R$, we introduce $u_r$ resources $r^1, \ldots, r^{u_r}$ to $R'$. We say that $r$ is the *representative* of $r^1, \ldots, r^{u_r}$. Then, a set $I' \subset R'$ is independent in $M_i'$ if the set $I$ that arises from $I'$ by replacing resources by their representatives is independent in $M_i$. This construction gives rise to a polynomial independence oracle for $M_i'$.

Now, we regard the matroid union $M' = M_1' \vee \cdots \vee M_n'$, see Definition A.1 in the Appendix, which again is a matroid. Using the algorithm proposed by Cunningham [6] we can compute a maximum-size set $\mathcal{B}$ in $I_1' \vee \cdots \vee I_n'$ in time polynomial in $n$, $m$, $rk(\mathcal{M})$, and the maximum complexity of the $n$ independence oracles.

Clearly, if $|\mathcal{B}| < \sum_{i \in N} \mathrm{rk}(M_i)$, there is no feasible packing of the bases of $M_1, \ldots, M_n$. If, in contrast, $|\mathcal{B}| = \sum_{i \in N} \mathrm{rk}(M_i)$, we obtain the corresponding strategies $(S_1, \ldots, S_n)$ using the algorithm. □

We now consider strategy spaces defined as *a-arborescences*, which are in general not matroids. Let $\mathcal{D} = (V, R)$ be a directed graph with $|R| = m$. For a distinguished node in $a \in V$, we define an *a-arborescence* as a directed spanning tree, where $a$ has in-degree zero and every other vertex has in-degree one.

**Theorem 2.5.** *The strategy packing problem can be solved in time $\mathcal{O}(m^2\, n^2)$ for a-arborescence games in which the set of strategies of each player equals the set of a-arborescences in a directed graph $\mathcal{D} = (V, R)$.*

*Proof.* The problem of finding $k$ disjoint $a$-arborescences in $G$ can be solved in time $\mathcal{O}(m^2\, k^2)$, see Gabow [10, Theorem 3.1]. Introducing $u_r$ copies for each edge $r \in R$, the problem of finding admissible strategies in the original problem is equivalent to finding $n$ disjoint $a$-arborescences. □

For single-commodity networks efficient computation of a SE is possible using well-known flow algorithms to implement the oracle. When we generalize to two commodities, however, a variety of problems concerning SE become NP-hard by a simple construction.

**Theorem 2.6.** *The strategy packing problem can be solved in time $\mathcal{O}(m^3)$ for single-commodity bottleneck congestion games.*

*Proof.* Assigning a capacity of $u_r$ to each edge and using the algorithm of Edmonds and Karp we obtain a maximum flow within $\mathcal{O}(m^3)$. Clearly, if the value of the flow is smaller than $n$, no admissible strategies exist and we can return $\emptyset$. If the flow is $n$ or larger we can decompose it in at least $n$ unit flows and return $n$ of them. □

**Theorem 2.7.** *In two-commodity network bottleneck games it is strongly NP-hard to (1) compute a SE, (2) decide for a given state whether any coalition has an improving move, and (3) decide for a given state and a given coalition if it has an improving move.*

*Proof.* We reduce from the 2 Directed Arc-Disjoint Paths (2DADP) problem, which is strongly NP-hard, see Fortune et al. [9]. The problem is to decide if for a given directed graph $\mathcal{D} = (V, A)$ and two node pairs $(s_1, t_1)$, $(s_2, t_2)$ there exist two arc-disjoint $(s_1, t_1)$- and $(s_2, t_2)$-paths. For the reduction, we define a corresponding two-commodity bottleneck game by introducing non-decreasing delay functions on every arc $r$ by $d_r(x) = 0$, if $x \leq 1$ and 1, else.

We associate every commodity with a player. Then, 2DADP is a Yes-instance if and only if every SE provides a payoff of zero to every player. For the other problems we simply construct a solution, in which the strategies are not arc-disjoint. The remaining results follow. □

# 3 Convergence of Improvement Dynamics

In the previous section, we have outlined some prominent classes of games, for which SE can be computed in polynomial time. Furthermore, it is known [11] that sequential improvement dynamics converge to PNE and SE. We now show that the Nash dynamics convergences quickly to a PNE in matroid games. For the convergence to SE one has to consider deviations of coalitions of players. However, deciding if such a deviation exists is NP-hard even in matroid games or single-commodity network games.

## 3.1 Matroid and Single-Commodity Network Games

We first observe that bottleneck congestion games can be transformed into regular congestion games while preserving useful properties regarding the convergence to PNE. This allows to show fast convergence to PNE in matroid bottleneck games.

### 3.1.1 Convergence to Pure Nash Equilibria

The following lemma establishes a connection between bottleneck and regular congestion games. For a bottleneck congestion game $G$ we denote by $G^{\text{sum}}$ the regular congestion game with the same congestion model as $G$ except that we choose $d'_r(S) = m^{d_r(\cdot)}, r \in R$.

**Lemma 3.1.** *Every PNE for $G^{sum}$ is a PNE for $G$.*

*Proof.* Suppose $S$ is a PNE for $G^{\text{sum}}$ but not for $G$. Thus, there is player $i \in N$ and strategy $S'_i \in \mathcal{S}_i$, such that $\max_{r\in S_i} d_r(\ell_r(S)) > \max_{r\in S'_i} d_r(\ell_r(S'_i, S_{-i}))$. We define $\bar{d} := \max_{r\in S'_i} d_r(\ell_r(S'_i, S_{-i}))$. This implies $\max_{r\in S_i} d_r(\ell_r(S)) \geq \bar{d} + 1$. We obtain a contradiction by observing

$$\sum_{r\in S_i} d'_r(\ell_r(S)) \geq \max_{r\in S_i} d'_r(\ell_r(S)) \geq m^{\bar{d}+1} > (m-1)\, m^{\bar{d}} \geq \sum_{r\in S'_i} d'_r(\ell_r(S'_i, S_{-i})) \quad .$$

□

We analyze the lazy best response dynamics considered for regular matroid congestion games presented in [1] and combine their analysis with Lemma 3.1. This allows to establish the following result.

**Theorem 3.2.** *Let $G$ be a matroid bottleneck congestion game. Then the lazy best response dynamics converges to a PNE in at most $n^2 \cdot m \cdot rk(\mathcal{M})$ steps.*

*Proof.* We consider the lazy best response dynamics in the corresponding game $G^{\text{sum}}$. In addition, we suppose that a player accepts a deviation only if his bottleneck value is strictly reduced. It follows that the duration is still bounded from above by $n^2 \cdot m \cdot \text{rk}(\mathcal{M})$ best responses as shown in [1]. □

### 3.1.2 Convergence to Strong Equilibria

For matroid bottleneck congestion games we have shown above that it is possible to converge to a PNE in polynomial time by a kind of best-response dynamics with unilateral improving moves. While previous work [11] establishes convergence to SE for every sequence of coalitional improving moves, it may already be hard to find one such move. In fact, we show that an $\alpha$-improving move can be strongly NP-hard to find, even if strategy spaces have simple matroid structures. This implies that deciding whether a given state is an $\alpha$-approximate SE is strongly co-NP-hard – even if all delay functions satisfy the $\beta$-bounded-jump condition[1] for any $\beta > \alpha$.

**Theorem 3.3.** *In matroid bottleneck congestion games it is strongly* NP*-hard to decide for a given state $S$ if there is some coalition $C \subseteq N$ that has an $\alpha$-improving move, for every polynomial time computable $\alpha$.*

*Proof.* We reduce from Set Packing. An instance of Set Packing is given by a set of elements $E$ and a set $\mathcal{U}$ of sets $U \subseteq E$, and a number $k$. The goal is to decide if there are $k$ mutually disjoint sets in $\mathcal{U}$. Given an instance of Set Packing we show how to construct a matroid game $G$ and a state $S$ such that there is an improving move for some coalition of players $C$ if and only if the instance of Set Packing has a solution.

The game will include $|N| = 1 + |\mathcal{U}| + |E| + \sum_{U \in \mathcal{U}} |U|$ many players. First, we introduce a master player $p_1$, which has two possible strategies. He can either pick a *coordination resource* $r_c$ or the *trigger resource* $r_t$. For each set $U \in \mathcal{U}$, there is a *set player* $p_U$. Player $p_U$ can choose either $r_t$ or a *set resource* $r_U$. For each set $U$ and each element $e \in U$, there is an *inclusion player* $p_{U,e}$. Player $p_{U,e}$ can use either the set resource $r_U$ or an *element resource* $r_e$. Finally, for each element $e$, there is an *element player* $p_e$ that has strategies $\{r_c, r_e\}$ and $\{r_c, r_a\}$ for some absorbing resource $r_a$.

The state $S$ is given as follows. Player $p_1$ is on $r_c$, all set players use $r_t$, all inclusion players the corresponding set resources $r_U$, and all element players the strategies $\{r_c, r_e\}$. The coordination resource $r_c$ is a bottleneck for the master player and all element players. The delays are $d_{r_c}(x) = \alpha + 1$, if $x > |E|$ and 1, otherwise. The trigger resource has delay $d_{r_t}(x) = 1$, if $x \leq |\mathcal{U}| - k + 1$, and $\alpha + 1$, otherwise. For the set resources $r_U$ the delay is $d_{r_U}(x) = 1$, if $x \leq 1$ and $\alpha + 1$, otherwise. Finally, for the element resources the delay is $d_{r_e}(x) = 1$ if $x \leq 1$ and $\alpha + 1$ otherwise.

Suppose that the underlying Set Packing instance is a Yes-instance, then an $\alpha$-improving move is as follows. The master player moves to $r_t$, the $k$ set players corresponding to a solution choose their set resources, the respective inclusion players move to the element resources, and all element players move to $r_a$. The delay of $r_c$ reduces from $\alpha + 1$ to 1, and the delay of $r_t$ reduces from $\alpha + 1$ to 1. Thus, the master player, all set players, and all element players improve their bottleneck by a factor of $\alpha + 1$. The migrating inclusion players do not interfere with each other on the element resources. Thus, they also improve the delay of their bottleneck resource by factor $\alpha + 1$, and we have constructed an $\alpha$-improving move for the coalition of all migrating players, all set players, and all element players.

Suppose that the underlying Set Packing instance is a No-instance. For contradiction, assume that there is a coalition $C$ that has an $\alpha$-improving move. Consider any player $p \in C$. We will show that for any player $p \neq p_1$, i.e., any set, inclusion, or element player, $p_1 \in C$ is

[1] Delay function $d_r$ satisfies the $\beta$-bounded-jump condition if $d_r(x+1) \leq \beta \cdot d_r(x)$ for any $x \geq 1$.

a prerequisite for achieving any strict improvement. We first note that the master player can never strictly improve without changing his strategy, because all element players will always use $r_c$ in their strategy. A move from $r_c$ to $r_t$ is an improvement if and only if at least $k$ set players drop $r_t$. These players must switch to the corresponding resources. However, for a set player $p_M$ such a move is an improvement if and only if all inclusion players on $r_U$ drop this resource from their strategy. These inclusion players must switch to the element resources. An inclusion player $p_{U,e}$ improves by such a move if and only if the element player drops the resource and $p_{U,e}$ is the only inclusion player moving to $r_e$. This implies that the moving set players must correspond to sets that are mutually disjoint. Finally, the element players move from $r_e$ to $r_a$ with delay $d_{r_a} = 0$, and this is an improvement if and only if the master player moves away from $r_c$. This last argument establishes that $p \in C$ implies $p_1 \in C$.

However, if the master player $p_1 \in C$, then we again follow the chain of reasoning above and see that the players corresponding to at least $k$ mutually disjoint sets must move and therefore be in $C$. This is a contradiction to having a No-instance.

Finally, we can add the resource $r_a$ to every strategy of the master, set, and inclusion players. In this way, the combinatorial structure of all strategy spaces is the same – a partition matroid $M$ with $rk(M) = 2$ and partitions of size 1 and 2 – only the mapping to resources is different for each player. □

The previous theorem shows hardness of the problem of finding a suitable coalition and a corresponding improving move. Even if we specify the coalition in advance and search only for strategies corresponding to an improving move, the problem remains strongly NP-hard.

**Corollary 3.4.** *In matroid bottleneck congestion games it is strongly* NP*-hard to decide for a given state $S$ and a given coalition $C \subseteq N$ if there is an $\alpha$-improving move for $C$, for every polynomial time computable $\alpha$.*

*Proof.* We will show this corollary using the games constructed in the previous proof by fixing the coalition $C = N$. Consider the construction in the previous proof. The coalition described above that has an improving move for a Yes-instance consists of the master player, all set players, all element players and the inclusion players that correspond to the sets of the solution to Set Packing. However, the inclusion players are only needed to transfer the chain of dependencies to the element players. We can set the strategy space of player $p_{U,e}$ to $\{r_h, r_l\} \times \{r_U, r_e\}$. Here $r_h$ and $r_l$ are two resources with delays $d_{r_h} = \alpha + 1$ and $d_{r_l} = 0$. In $S$ we assign the inclusion players to strategies $\{r_h, r_U\}$. Then an improving move for the inclusion players that remain on $r_U$ is to exchange $r_h$ by $r_l$. Thus, the problem of finding an arbitrary coalition with an improving move becomes trivial. However, we strive to obtain an improving move for $C = N$, and this must generate improvements for the master player and the set players. Thus, we still must reassign some inclusion players from the resources $r_U$ to the element resources $r_e$. Here we need to resolve conflicts as before, because otherwise inclusion players end up with a delay of $\alpha + 1$ on $r_e$ and do not improve. Following the previous reasoning we have an $\alpha$-improving move if and only if the underlying Set Packing instance is solvable. Finally, by appropriately adding dummy resources, we can again ensure that the combinatorial structure of all strategy spaces is the same. □

We can adjust the previous two hardness results on matroid games to hold also for single-commodity network games.

**Theorem 3.5.** *In single-commodity network bottleneck congestion games it is strongly* NP*-hard to decide for a given state $S$ (1) if there is some coalition $C \subseteq N$ that has an $\alpha$-improving move, and (2) if a given coalition $C \subseteq N$ has an $\alpha$-improving move, for every polynomial time computable $\alpha$.*

*Proof.* We transform the construction of Theorem 3.3 into a symmetric network bottleneck congestion game, see Fig. 1 for an example. First, we introduce for each resource $r_c$, $r_t$, $r_U$ for all $U \in \mathcal{U}$ and $r_e$ for all $e \in E$ an edge with the corresponding delay function as before. Additionally, we identify players and their strategies by routing them through a set of gadgets composed of edges, which have capacities implemented by cost functions that are 1 up to a capacity bound and $\alpha + 10$ above.

The first gadget is to separate the players into groups. An edge with capacity 1 identifies the master player, an edge with capacity $|\mathcal{U}|$ the set players, an edge with capacity $\sum_{U \in \mathcal{U}} |U|$ the inclusion players, and an edge with capacity $|E|$ the element players. The set and inclusion players are then further divided into their particular identities by edges of capacity 1. The element players route all over $r_c$. In addition, the master player has the alternative to route over $r_c$ or $r_t$. After the players have passed $r_c$ they again split into specific element players using edges of capacity 1. One player is allowed to route directly to the source $t$. This is meant to be the master player, but it does not hurt our argument if this is not the case.

After the players have routed through the capacitated gadgets, they can be assumed to reach an identification point (indicated by gray nodes in Fig. 1) and obtain an identity. Then they decide on a strategy from the previous game by routing over one of two allowed paths. In particular, we can allow the set players to route either over $r_t$ or their $r_U$, the inclusion players over $r_U$ or $r_e$, and the element players over $r_e$ or directly to the sink $t$.

We can create the corresponding state $S$ as before by assigning the master player to route over $r_c$ directly to the sink, the set players over $r_t$, the inclusion players over $r_U$ and the element players over $r_e$. This assignment is such that every player receives one identity (i.e., routes over exactly one gray node) and every identity is taken (i.e., every gray node is reached by exactly one player). This property also holds for every improving move – with the exception of one element player, who might route directly from $r_c$ to the sink, but as noted before this does not hurt the argument.

Our network structure allows to reconstruct the reasoning as before. Any improving move must include the master player, which improves if and only if he moves together with players corresponding to a solution to the Set Packing instance. Note that even by switching player identities, we cannot create an improving move when the underlying Set Packing instance is unsolvable. This proves the first part of the theorem.

For the second part, we use the same adjustment as in Corollary 3.4 to ensure that inclusion players can always improve. Directly before the middle fan out (see Figure 1) that results in identification of inclusion players we simply insert a small gadget with 2 parallel edges $r_l$ and $r_h$. In this way, all inclusion players must route over one of $r_l$ or $r_h$ and one of their corresponding $r_U$ or $r_e$. This resembles the strategy choices in the matroid game and yields hardness of computing an improving move for the coalition $C = N$. This proves the theorem. □

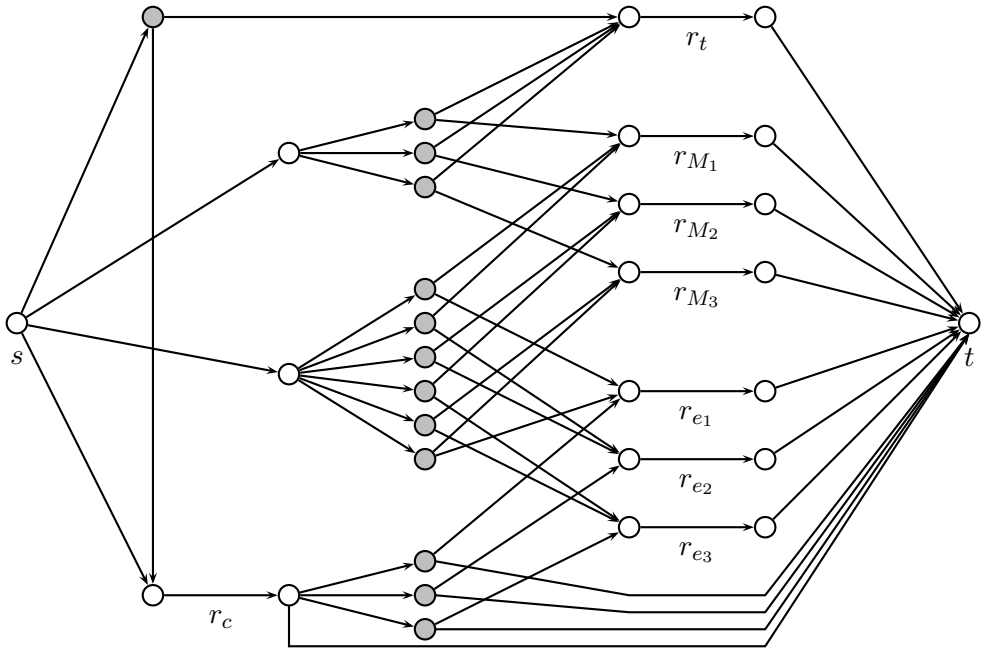


Figure 1: Network construction for a Set Packing instance with $\mathcal{U} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_3, e_1\}\}$. Gray nodes serve as identification for players as discussed in the text.

### 3.2 General Games and Approximation

The results of the previous sections imply hardness of the computation of SE or coalitional deviations, even in network games. Therefore, when considering general games we here restrict ourselves mostly to unilateral improving moves and PNE. Unfortunately, even in this restricted case the hardness results for regular congestion games in Skopalik and Vöcking [22] immediately imply identical results for bottleneck congestion games. The main result of [22] shows that computing an approximate PNE is PLS-hard. The proof is a reduction from CircuitFlip. We can regard the resulting congestion game as a bottleneck congestion game. It is straightforward to adjust all arguments in the proof of [22] to remain valid for bottleneck congestion games. We provide some details on the construction of the class $G(n)$ of games in the Appendix. A standard transformation [8] immediately yields the same result even for symmetric games, in which $\mathcal{S}_i = \mathcal{S}_j$ for all $i, j \in N$.

**Corollary 3.6.** *Finding an $\alpha$-approximate PNE in a symmetric bottleneck congestion game with positive and increasing delay functions is* PLS*-complete, for every polynomial-time computable $\alpha > 1$.*

A second result in [22] reveals that sequences of $\alpha$-improving moves do not reach an $\alpha$-approximate PNE quickly – even if all delay functions satisfy the $\beta$-bounded-jump condition with a constant $\beta$. Again, the proof remains valid if one regards the game as an asymmetric bottleneck congestion game. This yields the following corollary.

**Corollary 3.7.** *For every $\alpha > 2$, there is a $\beta > 1$ such that, for every $n \in \mathbb{N}$, there is a bottleneck congestion game $G(n)$ and a state $S$ with the following properties. The description length of $G(n)$ is polynomial in $n$. The length of every sequence of $\alpha$-improving moves leading from $S$ to an $\alpha$-approximate equilibrium is exponential in $n$. All delay functions of $G(n)$ satisfy the $\beta$-bounded-jump condition.*

Using the same trick as before to convert an asymmetric game in a symmetric one yields a similar result for symmetric games. However, we must sacrifice the $\beta$-bounded-jump condition of the delay functions, for every $\beta$ polynomial in $n$.

Despite the fact that (coalitional) improving moves are NP-hard to compute, one might hope that the state graph becomes sufficiently dense such that it allows short improvement paths. Unfortunately, we can show that this is not true, even if we consider all improving moves of coalitions of size up to $\mathcal{O}(n^{1-\epsilon})$, for any constant $\epsilon > 0$. Again, the same result holds for symmetric games when sacrificing the bounded-jump condition.

**Theorem 3.8.** *For every $\alpha > 2$, there is a $\beta > 1$ such that, for every $n \in \mathbb{N}$ and for every $k \in \mathbb{N}$, there is a bottleneck congestion game $G(n, k)$ and a state $S$ with the following properties. The description length of $G(n, k)$ is polynomial in $n$ and $k$. The length of every sequence of $\alpha$-improving moves of coalitions of size at most $k$ leading from $S'$ to an $\alpha$-approximate $k$-SE is exponential in $n$. All delay functions of $G(n, k)$ satisfy the $\beta$-bounded-jump condition.*

*Proof.* Our proof adjusts the construction of [22], which we recapitulate in the Appendix. The main idea of our adjustment is to construct a bottleneck congestion game $G(n, k)$ by generating $k$ copies of the game $G(n)$. We then add resources to the strategies. These resources make sure that there is a improvement step for a player in $G(n)$ if and only if there is a improvement step of corresponding $k$ players of the $k$ copies in $G(n, k)$.

To each strategy $j \in \{1, \ldots, 9\}$ of player every $\text{Main}_i$ of every the copy $m \in \{1, \ldots, k\}$ we add a resource $A^j_{i,k}$. Additionally, we add this resource to all strategies $j' \neq j$ of all players $\text{Main}_i$ of every other copy $m' \neq m$. Each of these resources has delay of $\delta^{i-1}$ if it is allocated by at most one player and $\delta^{i+3}$ otherwise. Analogously, we add resources to the strategies of the auxiliary players. That is, for every player $\text{Block}^j_i$ of every copy $m \in \{1, \ldots, k\}$, we add a resource $B^j_i$ in his strategy 1. We also add his resource in every strategy 2 of the every player $\text{Block}^j_i$ of every other copies $m' \neq m$. Similarly, for every player $\text{Block}^j_i$ of every copy $m \in \{1, \ldots, k\}$, we add a resource $C^j_i$ in his strategy 2, which we also add to every strategy 1 of the every player $\text{Block}^j_i$ of every other copies $m' \neq m$. Each of these resources has a delay of $\delta^{i-1}$ if it is allocated by at most one player and $\delta^{i+3}$ otherwise. Finally, we have to increase $\delta$ slightly.

We obtain the initial strategy profile $s'$ of $G(n, k)$ if every player of every copy $m$ of $G(n)$ plays according to the initial strategy profile $S$ of his copy. It it easy to see, that no coalition of less than $k$ player of a copy $m$ has an incentive to change their strategies. At least one of them would have to allocate a $A$-, $B$-, or $C$-resource that is already in use by another player. Thus, it is not an improvement step for these players. We, therefore, can conclude that all $k$ copies of a player always choose the same strategy. On the other hand, if there is an improving move of one player in $G(n)$, there is a coalitional improving move of all $k$ copies of that player in $G(n, k)$. If all players mimic this deviation in their copies, by construction, no two players allocate the same $A$-, $B$-, or $C$-resource. Furthermore, if the improvement step decreases the delay in $G(n)$, it does so for every copy of the player in $G(n, k)$.

Finally, note that as long as $k$ is polynomial in $n$ we obtain a reduction of polynomial size. In particular, for $k = n^{1/\epsilon - 1}$ we obtain a new game with $nk$ players, for which the unilateral moves of $G(n)$ are exactly moves of coalitions of size $(nk)^{1-\epsilon}$ and no smaller coalitions have improving moves. This proves the theorem. □

# 4 Conclusion

We have provided a detailed study of the computational complexity of exact and approximate pure Nash and strong equilibria in bottleneck congestion games. However, some important and fascinating open problems remain. While we have shown that results from [22] essentially translate, we were not able to establish the positive result of [4] about quick convergence to approximate PNE for symmetric games with bounded-jump delays. In addition, there are open problems regarding the duration of unilateral dynamics in symmetric network games and hardness of computing PNE in asymmetric networks. Finally, it would be interesting to see how results on centralized computation of SE extend to the computation of $\alpha$-approximate SE and $k$-SE, for $1 < k < n$.

# References


[1] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. *J. ACM*, 55(6), 2008.

[2] Robert Aumann. Acceptable points in general cooperative n-person games. In *Contributions to the Theory of Games IV*, volume 40 of *Annals of Mathematics Study*, pages 287–324. 1959.

[3] Ron Banner and Ariel Orda. Bottleneck routing games in communication networks. *IEEE J. Sel. Area Comm.*, 25(6):1173–1179, 2007.

[4] Steve Chien and Alistair Sinclair. Convergence to approximate Nash equilibria in congestion games. In *Proc. 18th Symp. Discrete Algorithms (SODA)*, pages 169–178, 2007.

[5] Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. Bottleneck links, variable demand, and the tragedy of the commons. In *Proc. 17th Symp. Discrete Algorithms (SODA)*, pages 668–677, 2006.

[6] William Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM J. Comput.*, 15(4):948–957, 1986.

[7] Jack Edmonds. Matroid partition. In G.B. Dantzig and A.F.Veinott, editors, *Mathematics of the Decision Sciences*, pages 335–345. AMS, 1968.

[8] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proc. 36th Symp. Theory of Computing (STOC)*, pages 604–612, 2004.

[9] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.

[10] Harold Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. Syst. Sci.*, 50(2):259–273, 1995.

[11] Tobias Harks, Max Klimm, and Rolf Möhring. Strong Nash equilibria in games with the lexicographical improvement property. In *Proc. 5th Intl. Workshop Internet & Network Economics (WINE)*, 2009.

[12] Ron Holzman and Nissan Law-Yone. Strong equilibrium in congestion games. *Games Econ. Behav.*, 21(1-2):85–101, 1997.

[13] S. Keshav. *An engineering approach to computer networking: ATM networks, the Internet, and the telephone network.* Addison-Wesley, 1997.

[14] Hideo Konishi, Michel Le Breton, and Shlomo Weber. Equilibria in a model with partial rivalry. *J. Econ. Theory*, 72(1):225–237, 1997.

[15] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer Verlag, 2002.

[16] Vladimir Mazalov, Burkhard Monien, Florian Schoppmann, and Karsten Tiemann. Wardrop equilibria and price of stability for bottleneck games with splittable traffic. In *Proc. 2nd Intl. Workshop Internet & Network Economics (WINE)*, pages 331–342, 2006.

[17] Crispin St. John Alvah Nash-Williams. An application of matroids to graph theory. In P. Rosenstiehl, editor, *Theory of Graphs; Proc. of an International Symposium in Rome 1966*, pages 263–265, 1967.

[18] Lili Qiu, Yang Richard Yang, Yin Zhang, and Scott Shenker. On selfish routing in internet-like environments. *IEEE/ACM Trans. Netw.*, 14(4):725–738, 2006.

[19] Robert Rosenthal. A class of games possessing pure-strategy Nash equilibria. *Intl. J. Game Theory*, 2:65–67, 1973.

[20] Tim Roughgarden. Routing games. In Noam Nisan, Éva Tardos, Tim Roughgarden, and Vijay Vazirani, editors, *Algorithmic Game Theory*, chapter 18. Cambridge University Press, 2007.

[21] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer Verlag, 2003.

[22] Alexander Skopalik and Berthold Vöcking. Inapproximability of pure Nash equilibria. In *Proc. 40th Symp. Theory of Computing (STOC)*, pages 355–364, 2008.

# Appendix

## A Basics in Matroid Theory

In the following, we will briefly introduce the notion of matroids. For a comprehensive introduction as well as for the proofs of the mentioned results we refer the reader to the textbooks of Korte and Vygen [15, Chapter 13] and Schrijver [21, Chapters 39 – 42].

Let $F$ be a finite set. A tuple $M = (F, \mathcal{I})$ where $\mathcal{I} \subset 2^F$ is called a *matroid* if $(i)$ $\emptyset \in \mathcal{I}$, $(ii)$ if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$, and $(iii)$ if $I, J \in \mathcal{I}$ and $|J| < |I|$, then there exists an $i \in I \setminus J$ with $J \cup \{i\} \in \mathcal{I}$. A set $A \subseteq F$ is called *independent* if $A \in \mathcal{I}$ and *dependent*, otherwise. The set of (inclusion wise) maximal independent subsets of $F$ is called the *basis* of $M$.

For given $F$, a matroid $(F, \mathcal{I})$ may be of exponential size, thus, one frequently assumes that a matroid comes with an *independence oracle* that returns for all sets $A \subseteq F$ whether $A \in \mathcal{I}$ or not. It shall be noted that for many subclasses of matroids an independence oracle can be implemented in polynomial time.

Another way of representing matroids is via a *rank function* $\mathrm{rk} : 2^F \to \mathbb{N}$. Every subcardinal, monotonic and sub-modular function rk gives rise to a matroid whose independent sets then are defined as $\{A \subseteq F : \mathrm{rk}(A) = |A|\}$. If the independent sets are known a priori via an independence oracle the rank function is defined as $\mathrm{rk}(A) = \max_{I \in \mathcal{I} : I \subseteq A} |I|$. With a slight abuse of notation, we define for a matroid $M = (F, \mathcal{I})$ the rank of the matroid itself as $\mathrm{rk}(M) = \mathrm{rk}(F)$.

To present our positive results for matroid bottleneck congestion games in a general framework we give the definition of matroid union. This concept has been introduced by Nash-Williams [17] and Edmonds [7].

**Definition A.1** (Matroid union)**.** *Let $M_1 = (S_1, \mathcal{I}_1), \ldots, M_k = (S_k, \mathcal{I}_k)$ be matroids. Define the union of these matroids as $M_1 \vee \cdots \vee M_k = (S_1 \cup \cdots \cup S_k, \mathcal{I}_1 \vee \cdots \vee \mathcal{I}_k)$ where*

$$\mathcal{I}_1 \vee \cdots \vee \mathcal{I}_k = \{I_1 \cup \cdots \cup I_k : I_1 \in \mathcal{I}_1, \ldots, I_k \in \mathcal{I}_k\}.$$

Nash-Williams proved that for $k$ matroids $M_1 = (S_1, \mathcal{I}_1), \ldots, M_k = (S_k, \mathcal{I}_k)$ their union $M_1 \vee \cdots \vee M_k$ is a matroid again. The maximum cardinality of an independent set in $\mathcal{I}_1 \vee \cdots \vee \mathcal{I}_k$ equals the maximum cardinality of a common independent set of two suitably constructed matroids. This observation reduces the problem of finding a maximum-size set in $\mathcal{I}_1 \vee \cdots \vee \mathcal{I}_k$ to the intersection problem of two matroids, which can be solved in polynomial time, see Cunningham [6].

## B Description of $G(n)$

In this section, we recapitulate the construction of $G(n)$ from [22]. This shows that (bottleneck) congestion games do not converge quickly to a PNE even if the players only perform unilateral $\alpha$-improving moves.

We construct a (bottleneck) congestion game $G(n)$ that resembles a recursive run of $n$ *programs*, i.e., sequences of unilateral $\alpha$-improving moves. After its activation, program $i$ triggers a run of program $i - 1$, waits until it finishes its run, and triggers it a second time.

These sequences are deterministic apart from the order in which some auxiliary players make their improvement steps.

| Strategies of $\text{Block}_i^j$ | Resources | Delays |
|---|---|---|
| (1) | $t_i^j$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^j$ | $\delta^{i-1}/\delta^{i+2}$ |
| (2) | $c_i^1$ | $2\alpha\delta^{i-1}/\delta^{i+2}$ |

Figure 2: Definition of the strategies of the players $\text{Block}_i^j$

A program $i$ is implemented by a gadget $G_i$ consisting of a main player that we call $\text{Main}_i$ and eight auxiliary players called $\text{Block}_i^1, \ldots, \text{Block}_i^8$. The main player has nine strategies numbered from 1 to 9. Each auxiliary player has two strategies, a first and a second one. A gadget $G_i$ is *idle* if all of its players play their first strategy. Gadget $G_{i+1}$ *activates* gadget $G_i$ by increasing the delay of (the bottleneck resource in) the first strategy of player $\text{Main}_i$. In the following sequence of improvement steps the player $\text{Main}_i$ successively changes to the strategies $2, \ldots, 8$. We call this sequence a *run* of $G_i$. During each run, $Main_i$ activates gadget $G_{i-1}$ twice by increasing the delay of the (bottleneck resource in the) first strategy of $\text{Main}_{i-1}$. Gadget $G_{i+1}$ is blocked (by player $\text{Block}_i^8$) until player $\text{Main}_i$ reaches its strategy 9. Then $G_{i+1}$ continues its run, that is, it decreases the delay of the bottleneck resource in the first strategy of player $\text{Main}_i$, waits until gadget $G_i$ becomes idle again, and afterwards triggers a second run of $G_i$. The role of the auxiliary players of $G_i$ is to control the strategy changes of $\text{Main}_i$ and $\text{Main}_{i+1}$.

In the initial state $s$, every gadget $G_i$ with $1 \leq i \leq n-1$ is idle. Gadget $G_n$ is activated. In every improvement path starting from $s$, gadget $G_i$ is activated $2^{n-i}$ times, which yields the theorem.

Now we go into the details of our construction. The (bottleneck) congestion game $G(n)$ consists of the gadgets $G_1, \ldots, G_n$. Each gadget $G_i$ consists of a player $\text{Main}_i$ and the players $\text{Block}_i^1, \ldots, \text{Block}_i^8$. The nine strategies of a player $\text{Main}_i$ are given in Figure 3. The two strategies of a player $\text{Block}_i^j$ are given in Figure 2. $\delta = 10\alpha^9$ is a scaling factor for the delay functions.

The auxiliary players implement a locking mechanism. The first strategy of player $\text{Block}_i^j$ is $\{t_i^j, b_i^j\}$ and its second strategy is $\{c_i^j\}$. The delays of the resources $b_i^j$ and $c_i^j$ are relatively small ($\delta^{i-1}$ and $2\alpha\delta^{i-1}$, respectively) if allocated by only one player. If they are allocated by two or more players, however, then each of them induce a significantly larger delay of $\delta^{i+2}$. Theses resources are also part of the strategies of $\text{Main}_i$ or $\text{Main}_{i+1}$. Note, that neither $\text{Main}_i$ nor $\text{Main}_{i+1}$ has an incentive to change to a strategy having a delay of $\delta^{i+2}$ or more. The delay of the resource $t_i^j$ is chosen such that $\text{Block}_i^j$ has an incentive to change to its second strategy if $\text{Main}_i$ allocates this resource. If $\text{Main}_i$ neither allocates this resource nor the resource $b_i^j$, it has an incentive to change to its first strategy. Due to scaling factor $\delta^{i-1}$ the delays of the resource $t_i^j$ do not affect the preferences of $\text{Main}_i$.

These definitions yield the following properties. If auxiliary player $\text{Block}_i^j$ of gadget $G_i$ plays its first strategy then this prevents $\text{Main}_i$ from choosing strategy $j+2$. Player $\text{Block}_i^j$ has an incentive to change to its second strategy only if player $\text{Main}_i$ chooses its strategy

$j + 1$. By this mechanism, we ensure that $\text{Main}_i$ chooses the strategies 1 to 8 in the right order. In addition, the first strategy of $\text{Block}_i^8$ prevents $\text{Main}_{i+1}$ from going to strategy 4 or 8. This ensures that $\text{Main}_{i+1}$ waits until the run of player $\text{Main}_i$ is completed. Furthermore, $\text{Main}_{i+1}$ can enter into strategy 3 or 7 only if all auxiliary players of gadget $G_i$ use their first strategy. This ensures that a run starts with all auxiliary players being in their first strategy.

This shows that in every sequence of improvement steps from $s$ to a Nash equilibrium in the (bottleneck) congestion game $G(n)$ each gadget $i$ is activated $2^{n-i}$ times. One can easily check that every improvement step of a player decreases its delay (of the bottleneck resource) by a factor of at least $\alpha$ and every delay function satisfies the $\beta$-bounded-jump condition with $\beta = \delta^3$ with $\delta = 10\alpha^9$.

| Strategy | Resources | Delays |
|---|---|---|
| (1) | $e_i^1$ | $\delta^i/9\alpha^9\delta^i$ |
| (2) | $e_i^2$ | $8\alpha^8\delta^i$ |
| | $c_{i-1}^1, \ldots, c_{i-1}^9$ | $2\alpha\delta^{i-2}/\delta^{i+1}$ |
| | $t_i^1$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| (3) | $e_i^3$ | $7\alpha^7\delta^i$ |
| | $e_{i-1}^1$ | $\delta^{i-1}/9\alpha^9\delta^{i-1}$ |
| | $t_i^2$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^1$ | $\delta^{i-1}/\delta^{i+2}$ |
| (4) | $e_i^4$ | $6\alpha^6\delta^i$ |
| | $b_{i-1}^8$ | $\delta^{i-2}/\delta^{i+1}$ |
| | $t_i^3$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^2$ | $\delta^{i-1}/\delta^{i+2}$ |
| (5) | $e_i^5$ | $5\alpha^5\delta^i$ |
| | $t_i^4$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^3$ | $\delta^{i-1}/\delta^{i+2}$ |
| (6) | $e_i^6$ | $4\alpha^4\delta^i$ |
| | $c_{i-1}^1, \ldots, c_{i-1}^9$ | $2\alpha\delta^{i-2}/\delta^{i+1}$ |
| | $t_i^5$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^4$ | $\delta^{i-1}/\delta^{i+2}$ |
| (7) | $e_i^7$ | $3\alpha^3\delta^i$ |
| | $e_{i-1}^1$ | $\delta^{i-1}/9\alpha^9\delta^{i-1}$ |
| | $t_i^6$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^5$ | $\delta^{i-1}/\delta^{i+2}$ |
| (8) | $e_i^8$ | $2\alpha^2\delta^i$ |
| | $b_{i-1}^8$ | $\alpha\delta^{i-2}/\delta^{i+1}$ |
| | $t_i^7$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^6$ | $\delta^{i-1}/\delta^{i+2}$ |
| (9) | $e^9$ | $\alpha\delta^i$ |
| | $t_i^8$ | $\delta^{i-1}/2\alpha^2\delta^{i-1}$ |
| | $b_i^7$ | $\delta^{i-1}/\delta^{i+2}$ |

Figure 3: Definition of the strategies of the players $\text{Main}_i$. The delay of resource $e_n^1$ is constantly $9\alpha^9\delta^n$.