A logarithmic approximation for polymatroid congestion games

Tobias Harks, Tim Oosterwijk, Tjark Vredeveld

Maastricht University, Tongersestraat 53, 6211 LM, Maastricht, The Netherlands

Abstract

We study the problem of computing a social optimum (minimum cost solution) in polymatroid congestion games, where the strategy space of every player consists of the set of vectors in a player-specific integral polymatroid base polyhedron defined on the ground set of resources. For general non-decreasing cost functions we devise an H_{rk}-approximation algorithm, where rk is the sum of the ranks of the player-specific polymatroids and H_{rk} denotes the rk-th harmonic number. The main idea of our algorithm is to iteratively increase resource utilization in a greedy fashion and to invoke a polynomial covering oracle that checks feasibility of every computed resource utilization. The approximation guarantee is best possible up to a constant factor. As a special case, our result (partially) settles an open problem of Ackermann et al. (H. Ackermann, H. Röglin, and B. Vöcking. On the impact of combinatorial structure on congestion games. *J. ACM*, 55(6):1–22, 2008. Section 2.2) where the complexity of computing a socially optimal solution for matroid congestion games with non-decreasing cost functions is considered. Here, the approximation guarantee is best possible up to a constant factor if the number of resources is polynomially bounded in the number of players.

Keywords: Congestion game, Approximation Algorithm, Polymatroid, Matroid

1. Introduction

Congestion games have become a standard game-theoretic model describing the allocation of exhaustible resources by selfish players. In the basic model of Rosenthal [19], there is a finite set of players and resources and each player is associated with a set of allowable subsets of resources. A pure strategy of a player consists of an allowable subset. Congestion on a resource is modeled by a load-dependent cost function which is usually non-decreasing and solely depends on the number of players using the resource. In the context of *network games*, the resources may correspond to edges of a graph, the allowable subsets correspond to the simple paths connecting a source and a sink and players choose minimum cost paths. Rosenthal proved in his seminal paper that congestion games always admit a pure Nash equilibrium.

In this paper, we focus on so-called *polymatroid congestion games*, where the strategy space of every player consists of the set of vectors in a player-specific integral polymatroid base polyhedron defined on the ground set of resources. This can be viewed as a game in which every player chooses a multiset of the resources rather than a subset. These games have numerous applications as they include for instance matroid congestion games, singleton congestion games (uniform rank-1 matroids) or spanning tree congestion games, where every player selects a spanning tree of a player-specific subgraph of a given graph. We consider the problem of computing a minimum cost solution in polymatroid congestion games. This problem is important for scenarios where a central planner can implement a solution or when players collaborate. Additionally, minimum cost solutions serve as building blocks for other cost-efficient solutions, e.g., as in [25], where a minimum

Email address: {t.harks,t.oosterwijk,t.vredeveld}@maastrichtuniversity.nl ()

cost solution is used for defining cost sharing protocols with low price of stability/anarchy. In fact, Ackermann et al. [1, Section 2.2.] as well as von Falkenhausen and Harks [25, Section 5] state as an open problem to characterize the computational complexity of computing a minimum cost solution in matroid congestion games with non-decreasing cost functions, which is a special case of the problem we consider in this paper (cf. Remark 2).

1.1. Our Results

We devise an H_{rk}-approximation algorithm, where rk is the sum of the ranks of the player-specific polymatroids (the value of the polymatroid function on the entire set of resources) and H_{rk} denotes the rk-th harmonic number. This approximation guarantee is best possible (up to a constant factor) for algorithms with polynomial running time, unless NP $\subseteq$ TIME($n^{\mathcal{O}(\log\log n)}$). As a byproduct we show that matroid congestion games are also H_{rk}-approximable in polynomial time (cf. Remark 2), a result that partially settles an open problem of Ackermann et al. [1, Section 2.2.] as well as von Falkenhausen and Harks [25, Section 5]. For matroid congestion games, we only leave a gap if rk is not polynomially bounded in n.

Our algorithm maintains data structures for *target loads* and *preliminary cost-per-unit values* for every resource and its respective load. The algorithm iteratively increases the target loads on the resources by selecting the resource (with corresponding target load) having the lowest cost per unit. After this greedy choice, a covering oracle is invoked that checks whether or not there exists a feasible strategy profile (a vector in the sum of the player-specific integral polymatroid base polyhedra) covering the currently computed target loads. If the covering oracle returns a feasible strategy profile, we update target loads and preliminary cost-per-unit values and proceed. If the oracle returns infeasibility, we reduce the maximum target load for the selected resource and proceed. Denoting the number of resources by m, pseudopolynomial running time in terms of oracle calls follows as there are only $m\,\mathrm{rk}$ possible target loads and in each iteration a target load is increased or a maximum target load is decreased. If rk is polynomial in the input size (as is the case for matroids, arborescences, etc.) the algorithm runs in polynomial time. The oracle itself can also be implemented in linear oracle time using an adaptation of the polymatroid greedy algorithm.

1.2. Literature review

Computing a social optimum. Werneck et al. [26] studied the complexity of computing a social optimum in spanning tree congestion games. For convex cost functions they devised an efficient algorithm computing an optimal solution. Essentially, convex cost functions allow to linearize the cost function and then to apply a greedy algorithm. Ackermann et al. [1] extended the work of [26] by observing that the same idea is applicable to matroid congestion games still requiring that cost functions need to be convex. For spanning tree games with non-monotonic cost functions, they showed that computing a social optimum is NP-hard. The case of matroid congestion games with general non-decreasing cost functions is posed as an open problem [1, Section 2.2.].

It should be noted that the positive results for convex cost functions were already implied by previous works, perhaps not so obvious. Groenevelt [12] and Fujishige [9] presented polynomial time algorithms to minimize a convex separable function over an integral polymatroid base polyhedron. Since the matroid rank function is submodular, the strategy space for every player can equivalently be represented as an integral polymatroid base polyhedron. Using that the sum of polymatroid base polyhedra is again a polymatroid base polyhedron, the results of Groenevelt [12] and Fujishige [9] thus already imply a polynomial time algorithm for computing a social optimum for matroid congestion games with convex cost functions. For polymatroids with fixed costs for all resources, Wolsey [27] showed that the greedy algorithm gives a logarithmic approximation. In contrast to these works, we consider the case of arbitrary non-decreasing cost functions.

A special case of polymatroid congestion games is that of singleton congestion games (uniform rank-1 matroids) with arbitrary non-decreasing cost functions. Harks and von Falkenhausen [15] devised an H_n-approximation approximation algorithm for the social cost, where n is the number

of players. Their algorithm is based on successive network flow computations on a suitably defined capacitated graph. Moreover, they showed this result is essentially best possible up to a constant factor, as they show that this optimization problem is hard to approximate within a factor of $c \log n$ for any $c < 1$, a reduction we will extend for our hardness result (cf. Lemma 1).

Meyers and Schulz [17] classified the complexity of computing a social optimum for general congestion games as well as network congestion games and differentiated between asymmetric and symmetric strategy spaces. In the case of network congestion games, they also distinguished the case in which all players share a common source. Regarding the cost functions, they differentiated between five types: non-decreasing, convex non-decreasing, non-increasing, concave non-increasing, and non-monotonic cost functions. For all combinations of strategy spaces and cost functions they established the complexity of finding the social optimum. Most of the resulting problems are inapproximable to any finite factor. In particular, the asymmetric case with non-decreasing costs is not approximable to any finite factor. Very recently, Roughgarden [20] studied the impact of the computational complexity of computing socially optimal solutions on the price of anarchy. He derived a reduction that translates inapproximability results to corresponding lower bounds on the price of anarchy. In the context of congestion games, he derived stronger inapproximability bounds for Rosenthal's congestion model involving polynomial cost functions (with non-negative coefficients).

Computing a socially optimal profile has also been studied in the congestion model of Milchtaich [18], where resource costs are player-specific. Chakrabarty et al. [4] proved that the social optimum is inapproximable within any finite factor, unless $P = NP$. They exhibited some special cases in which a minimum cost solution can be found in polynomial time, e.g. when the number of strategies is bounded. Blumrosen and Dobzinski [3] considered the problem of maximizing welfare instead of minimizing costs and presented an 18-approximation for this problem. Assuming non-decreasing cost functions, they improved the approximation guarantee to $\frac{e}{e-1}$. De Keijzer and Schäfer [6] studied congestion games with positive externalities, where players benefit from other players choosing the same resource. They showed even very special cases of the problem are NP-hard and provided several approximation algorithms.

Computing equilibria. In the past decade the computational complexity to find a pure Nash equilibrium (PNE) in congestion games has been studied extensively. Ackermann et al. [1] proved that for congestion games with non-decreasing cost functions, matroids are the maximal property on the strategy space of every player that guarantees that best responses for players converge to a PNE in polynomial time. Fabrikant et al. [7] showed that a PNE can be found in polynomial time in symmetric network congestion games with non-decreasing cost functions. However, for general network games with non-decreasing cost functions, finding a PNE is PLS-complete [7]. These results have been strengthened to hold even when the cost functions are non-decreasing and linear [1]. Also finding α-approximate PNEs in congestion games is PLS-complete for any $\alpha > 1$ [22].

If the strategy space is restricted to symmetric singleton congestion games, the best Nash equilibrium can be found in polynomial time for any cost function [16]. Sperber [23] showed that both the best and the worst PNE can be found in polynomial time for non-decreasing cost functions with a greedy algorithm. However, she proved that on series-parallel graphs this is NP-hard, except in the case of finding the best PNE in a 2-player game. Also the setting in which the social cost is not determined by the sum of the costs but by the maximum cost (*makespan* social cost) has been studied. In this setting, the worst PNE can be found in series-parallel graphs, however, finding the best PNE is NP-complete [11]. Another related class of games are so-called *bottleneck congestion games*, in which the total cost of a player is not the sum but the maximum of the costs of the resources he chose. Harks et al. [13] devised an algorithm computing PNEs (even strong equilibria) which relies on the idea of a *strategy packing oracle*, which is similar to the strategy covering oracle in this paper. They also maintain target loads that are iteratively updated, and they also use that for matroids, this strategy packing oracle can be implemented in polynomial time.

2. The Model

2.1. Congestion Models

A *congestion model* is given by the tuple $\mathcal{M} = (N, R, \mathcal{B}, (c_r)_{r\in R})$, where $N = \{1, \ldots, n\}$ is a non-empty, finite set of players and $R = \{1, \ldots, m\}$ is a non-empty, finite set of *resources*. Every player $i \in N$ chooses a *pure strategy* $\mathbf{b}^i \in \mathbb{N}^R$ from a non-empty bounded integral polyhedron $\mathcal{B}_i \subseteq \mathbb{N}^R$ (which can be seen as a non-empty, finite collection of multisets). We denote the set of joint strategy profiles by the Minkowski sum of the polyhedra $\mathcal{B} = \mathcal{B}_1 + \ldots + \mathcal{B}_n$. Given a strategy profile $\mathbf{b} \in \mathcal{B}$, we define the *load* of a resource $r \in R$ as b_r, which is the r-th component of the vector $\mathbf{b}$. All resources $r \in R$ have a load-dependent *cost function* $c_r : \mathbb{N} \to \mathbb{N}$. Abusing notation, we write $c_r(\mathbf{b}) = c_r(b_r)$ for all $\mathbf{b} \in \mathcal{B}$. We assume for all $r \in R$ that $c_r(0) = 0$ and $c_r(i) \geq c_r(j)$ whenever $i \geq j$. We denote the *social cost* by $C(\mathbf{b}) = \sum_{r\in R} c_r(\mathbf{b})$.

2.2. Polymatroid Congestion Models

A function $f : 2^R \to \mathbb{N}$ is called submodular if $f(U) + f(V) \geq f(U \cup V) + f(U \cap V)$ for all $U, V \subseteq R$. It is called monotone if $f(U) \leq f(V)$ for all $U \subseteq V$, and normalized if $f(\emptyset) = 0$. A pair (R, f) is an *integral polymatroid* if $f : 2^R \to \mathbb{N}$ is submodular, monotone and normalized. f is then called an *integral polymatroid rank function* and the associated integral polyhedron is defined as

$$\mathbb{P}_f := \left\{\mathbf{b} \in \mathbb{N}^R \mid b(U) \leq f(U)\ \forall U \subseteq R\right\},$$

where we define the load on a set U of resources as $b(U) = \sum_{r\in U} b_r$ (note that $b(\emptyset) = 0$). Given the integral polyhedron $\mathbb{P}_f$ and the integer $\mathrm{rk} = f(R)$, which we refer to as the *rank* of the polymatroid, the corresponding *integral polymatroid base polyhedron* is

$$\mathbb{B}_f(\mathrm{rk}) := \left\{\mathbf{b} \in \mathbb{N}^R \mid b(U) \leq f(U)\ \forall U \subseteq R,\ b(R) = \mathrm{rk}\right\}.$$

To obtain a polymatroid congestion model, we associate an integral polymatroid rank function f_i with every player $i \in N$, we denote $f_i(R) = \mathrm{rk}_i$ and let $\mathrm{rk} = \sum_{i\in N} \mathrm{rk}_i$. We denote the Minkowski sum of the polyhedra by $\mathbb{B}_f(\mathrm{rk}) := \sum_{i\in N} \mathbb{B}_{f_i}(\mathrm{rk}_i)$ and from [21, Theorem 44.6] we know that this is also an integral polymatroid base polyhedron.

We define a *polymatroid congestion model* $\mathcal{M} = (N, R, \mathbb{B}_f(\mathrm{rk}), (c_r)_{r\in R})$ as a congestion model, where every player $i \in N$ chooses an element $\mathbf{b}^i \in \mathbb{B}_{f_i}(\mathrm{rk}_i)$. In this polymatroid congestion model we want to find a vector $\mathbf{b} = \mathbf{b}^1 + \ldots + \mathbf{b}^n \in \mathbb{B}_f(\mathrm{rk})$ that minimizes the normalized monotone cost function $C(\mathbf{b}) = \sum_{r\in R} c_r(\mathbf{b})$.

We remark that polymatroid congestion games were recently introduced by Harks et al. [14]. In contrast to their model, we do not allow that cost functions are player-specific, but we do allow general non-decreasing cost functions instead of convex cost functions.

2.3. Computing Minimum Cost Strategy Profiles

We study the problem of computing an *optimal strategy profile* minimizing the social cost. Formally, we arrive at the following combinatorial optimization problem.

Problem 1. *Find Optimal Strategy Profile*

Input: A polymatroid congestion model $\mathcal{M} = (N, R, \mathbb{B}_f(\mathrm{rk}), (c_r)_{r\in R})$.

Objective: Find $\min_{\mathbf{b}\in\mathbb{B}_f(\mathrm{rk})} C(\mathbf{b})$.

3. A logarithmic approximation

Before we present our approximation algorithm, we derive the following hardness result. The reduction is based on [15, Theorem 7.1], where the hardness of computing an optimal strategy profile for singleton congestion games is shown.

Lemma 1. *Problem 1 is strongly NP-complete and there are no $c \log rk$ approximation algorithms for any $c < 1$, unless NP $\subseteq$ TIME$(n^{\mathcal{O}(\log \log n)})$.*

Proof. We reduce from the Hitting Set problem. An instance of Hitting Set consists of a set $\mathcal{C}$ of n subsets $(C_i)_{i \in N}$ over a finite ground set of elements E. A hitting set is a subset $F \subseteq E$ such that F contains at least one element of every $C_i \in \mathcal{C}$. The goal is to select a minimum cardinality hitting set.

Given an instance $(E, \mathcal{C})$ of Hitting Set, we construct a polymatroid congestion game $\mathcal{M}$ as follows. First construct the game $\mathcal{M}'$ by identifying E with R, and defining the submodular, monotone, normalized function f_i for all $i \in N$ as follows: $f_i(S) = 1$ if $S \cap C_i \neq \emptyset$ and $f_i(S) = 0$ otherwise. We let $c_r(0) = 0$ and $c_r(j) = 1$ for all $r \in R$ and $j \in N$.

From this game $\mathcal{M}'$, we construct another polymatroid congestion game $\mathcal{M}$ with one player, whose integral polymatroid base polyhedron is the Minkowski sum of the polyhedra in $\mathcal{M}'$, i.e. $\mathbb{B}_f(\mathrm{rk}) = \sum_{i \in N} \mathbb{B}_{f_i}(\mathrm{rk}_i)$. Note that $\mathrm{rk}_1 = \mathrm{rk} = n$.

By construction, there is a hitting set of cardinality k if and only if there is a vector $\mathbf{b} \in \mathbb{B}_f(\mathrm{rk})$ with $C(\mathbf{b}) \leq k$ in $\mathcal{M}$. Therefore, if we can approximate the social optimum in $\mathcal{M}$ within a factor of $c \log(\mathrm{rk})$ for some constant c, we can approximate the Hitting Set instance within $c \log(n)$. The lemma now follows from the fact that the Hitting Set problem is equivalent to the Set Cover problem [2], for which there are no polynomial time $c \log n$-approximation algorithm for any $c < 1$ unless NP $\subseteq$ TIME$(n^{\mathcal{O}(\log \log n)})$ [8]. □

Now we present our algorithm (see Algorithm 1). Intuitively, for every player i we want to distribute rk_i units over the resources such that the resulting multiset of resources corresponds to a vector $\mathbf{b}^i \in \mathbb{B}_{f_i}(\mathrm{rk}_i)$. Overall, this leads to a distribution of rk units over the resources such that $\mathbf{b} = \mathbf{b}^1 + \ldots + \mathbf{b}^n \in \mathbb{B}_f(\mathrm{rk})$. The idea is to incrementally increase the number of units distributed over the resources in a greedy fashion. Initially, we start with an empty distribution, i.e., the load on every resource is zero. Then, we will iteratively increase the load on some resource $r \in R$ to some desired *target load* denoted by $t_r \in \mathbb{N}$ (initially $t_r = 0$ for all $r \in R$). To make sure that the sum of the target loads does not exceed rk, we set upper bounds $t_r^{\max}$ on the target loads that are initially set to $t_r^{\max} = \mathrm{rk}$ for all $r \in R$. In every iteration, we select a resource $r \in R$ and a target load between $t_r + 1$ and $t_r^{\max}$ minimizing the *cost per unit*, which is defined via the following function:

$$h : R \times ([t_r + 1, t_r^{\max}] \cap \mathbb{N}) \to \mathbb{R}; \quad h(r, j) = \frac{c_r(j) - c_r(t_r)}{j - t_r}. \tag{1}$$

Let (r^*, j^*) be a minimizer of h. To check whether or not it is possible to distribute $j^* - t_{r^*}$ additional units on resource r^*, or equivalently, to increase the target load t_{r^*} to j^*, we call a *strategy covering oracle* denoted by $\mathfrak{O}(R, \mathcal{B}, (t_r)_{r \in R})$ (cf. Definition 1). This oracle checks if there is a strategy profile covering the current target loads, that is, if there is a $\mathbf{b} = \mathbf{b}^1 + \ldots + \mathbf{b}^n \in \mathbb{B}_f(\mathrm{rk})$ satisfying $b_r \geq t_r$ for all $r \in R$. It is similar to the strategy packing oracle from [13].

Throughout the paper, for two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{N}^m$, we use $\mathbf{x} \geq \mathbf{y}$ in the sense $x_i \geq y_i$ for all i, and for convenience we write $x(\{1, \ldots, i\}) = \sum_{j=1}^{i} x_j$.

Definition 1. *Strategy covering oracle* $\mathfrak{O}(R, \mathcal{B}, (t_r)_{r \in R})$

Input: A finite set R with target loads $t_r \in \mathbb{N}$ for all $r \in R$ and an integral polyhedron $\mathcal{B} \subseteq \mathbb{N}^R$.

Output: A vector $\mathbf{b} \in \mathcal{B}$ such that $\mathbf{b} \geq \mathbf{t}$, or the information that no such vector exists.

If the answer of the oracle is negative, it is not possible to increase the load on resource r^* to j^*. We set t_{r^*} back to its old value and update $t_{r^*}^{\max}$ to $j^* - 1$, because this target load j^* was too high for this resource. On the other hand, if the answer is affirmative, we keep the increased target load j^* on r^*, and for all $r \in R$ we update $t_r^{\max}$ to $\min\left\{t_r^{\max}, t_r + \mathrm{rk} - \sum_{r' \in R} t_{r'}\right\}$ to avoid target loads whose sum exceeds rk. We also update the values for h according to Equation (1). We then continue this procedure by finding a new minimizer of h and calling the oracle again. Note that in every iteration we increase the lower bound t_{r^*} or decrease the upper bound $t_{r^*}^{\max}$, thus, the domain of h becomes strictly smaller in every iteration. Indeed, once we have $\sum_{r \in R} t_r = \mathrm{rk}$ and the oracle outputs a vector $\mathbf{b} \in \mathbb{B}_f(\mathrm{rk})$ that meet these target loads, the algorithm terminates. The vector $\mathbf{b}$ can be decomposed in a vector for every player. For a formal description see Algorithm 1.

Algorithm 1: Algorithm for the polymatroid congestion model and game

Input: A polymatroid congestion model $\mathcal{M} = (N, R, \mathbb{B}_f(\mathrm{rk}), (c_r)_{r \in R})$
Output: A vector $\mathbf{b} \in \mathbb{B}_f(\mathrm{rk})$

$\mathbf{b} \leftarrow \mathbf{0}$;
$t_r \leftarrow 0$ and $t_r^{\max} \leftarrow \mathrm{rk}$ for all $r \in R$;
while $\sum_{r \in R} t_r < \mathrm{rk}$ **do**
  Choose $(r^*, j^*) \in \operatorname{argmin}_{r \in R, j \in [t_r + 1, t_r^{\max}]} h(r, j)$;
  $temp \leftarrow t_{r^*}$;
  $t_{r^*} \leftarrow j^*$;
  if $\mathfrak{O}(R, \mathcal{B}, (t_r)_{r \in R}) = \mathbf{b}'$ **then**
    $\mathbf{b} \leftarrow \mathbf{b}'$;
    $t_r^{\max} \leftarrow \min\left\{t_r^{\max}, t_r + \mathrm{rk} - \sum_{r' \in R} t_{r'}\right\}$ for all $r \in R$;
    Update $h(r^*, j)$ for all $j \in [t_{r^*} + 1, t_{r^*}^{\max}]$ as in Equation (1);
  else
    $t_{r^*} \leftarrow temp$;
    $t_{r^*}^{\max} \leftarrow j^* - 1$;
return b;

The following lemma shows that the strategy covering oracle can be implemented in linear oracle time for our polymatroid congestion model.

Lemma 2. *If $\mathcal{B}$ is an integral polymatroid base polyhedron, then the strategy covering oracle can be implemented in time $\mathcal{O}(m\,Q)$, where Q is the complexity of the value giving oracle that returns $f(U)$ for any $U \subseteq R$.*

Proof. See Algorithm 2 for a formal description, which is an adaptation of the polymatroid greedy algorithm (see e.g. [21, Theorem 44.3]).

There exists a vector $\mathbf{b} \in \mathbb{B}_f(\mathrm{rk})$ such that $\mathbf{b} \geq \mathbf{t}$ if and only if $\mathbf{t} \in \mathbb{P}_f(\mathrm{rk}) = \sum_{i=1}^n \mathbb{P}_{f_i}(\mathrm{rk}_i)$. To check this membership, we start at $\mathbf{y} = \mathbf{0} \in \mathbb{P}_f(\mathrm{rk})$ and iteratively increase y_i to t_{r_i}. Following the proof of [21, Theorem 44.3] we know that after iteration i, $\mathbf{y} \in \mathbb{P}_f(\mathrm{rk})$ if and only if $y(\{1, \ldots, i\}) \leq f(\{1, \ldots, i\})$. After m iterations $\mathbf{y} = \mathbf{t}$, hence $\mathbf{t} \in \mathbb{P}_f(\mathrm{rk})$ and the target loads are feasible. If at some point $y(\{1, \ldots, i\}) > f(\{1, \ldots, i\})$, the target loads are infeasible.

To extend the target loads to a strategy profile (i.e. a vector $\mathbf{b} \in \mathbb{B}_f(\mathrm{rk})$), we define the function $g(U) = f(U) - t(U)$ for all $U \subseteq R$ (for the implementation of the algorithm we only need $g(\{1, \ldots, i\})$ for all i). We claim, and this is easy to check, that g is an integral polymatroid rank function with $g(R) = \mathrm{rk} - t(R)$. Hence we can find a vector $\mathbf{z} \in \mathbb{B}_g(g(R))$ using the polymatroid

greedy algorithm. Define $\mathbf{b} = \mathbf{t} + \mathbf{z}$. By construction $b(U) = t(U) + z(U) \leq t(U) + f(U) - t(U) = f(U)$ for all $U \subseteq R$, so $\mathbf{b} \in \mathbb{P}_f(\mathrm{rk})$. Similarly, $b(R) = \mathrm{rk}$ and hence $\mathbf{b} \in \mathbb{B}_f(\mathrm{rk})$. As $\mathbf{z} \geq \mathbf{0}$ we have $\mathbf{b} \geq \mathbf{t}$ and the proof is complete.

Denoting the complexity of the value giving oracle for the function f by Q, the running time of both the membership check and the extension is $\mathcal{O}(m\,Q)$. □

Algorithm 2: Strategy covering oracle for polymatroid congestion games

Input: An integral polymatroid base polyhedron $\mathcal{B}$ and a vector t
Output: A vector $\mathbf{b} \in \mathcal{B}$ such that $\mathbf{b} \geq t$, or the information that no such vector exists.

$\mathbf{y} \leftarrow \mathbf{0}$;
for $i = 1$ **to** m **do**
  $y_i \leftarrow t_{r_i}$;
  if $y(\{1, \ldots, i\}) > f(\{1, \ldots, i\})$ **then**
    return "Infeasible target loads"
$\mathbf{z} \leftarrow \mathbf{0}$ and $g(\emptyset) \leftarrow 0$;
for $i = 1$ **to** m **do**
  $g(\{1, \ldots, i\}) \leftarrow f(\{1, \ldots, i\}) - t(\{1, \ldots, i\})$;
  $z_i \leftarrow g(\{1, \ldots, i\}) - g(\{1, \ldots, i-1\})$;
return $\mathbf{t} + \mathbf{z}$;

We now prove the following theorem.

Theorem 1. *Algorithm 1 is an H_{rk}-approximation algorithm for Problem 1 that runs in time $\mathcal{O}(m^2\,\mathrm{rk}^2\,Q)$, where Q is the complexity of the value giving oracle that returns $f(U)$ for any $U \subseteq R$.*

We show this using the following two lemmata. For the proof of the next lemma, we need the following well-known property of polymatroids (cf. [9, Theorem 3.27] and the text following [21, Theorem 44.6]).

Property 1. *Let $\mathbb{P}_f(\mathrm{rk})$ be an integral polymatroid polyhedron. For any two vectors $\mathbf{b}, \mathbf{b}' \in \mathbb{P}_f(\mathrm{rk})$ and any element $r \in R$ with $b_r > b'_r$, there exists an element $r' \in R$ with $b'_{r'} > b_{r'}$ such that $\mathbf{b} - \chi_r + \chi_{r'} \in \mathbb{P}_f(\mathrm{rk})$. Here, χ_r is the m-dimensional unit vector corresponding to resource $r \in R$.*

Lemma 3. *Let $\mathbf{b}$ be the output of Algorithm 1 and let $\mathbf{b}^*$ be the set of bases minimizing $C(\mathbf{b}^*)$. Then $C(\mathbf{b}) \leq H_{\mathrm{rk}} C(\mathbf{b}^*)$.*

Proof. Consider iteration k of Algorithm 1, indexed by the while loop. We denote the target load of a resource r at the start of this iteration by t_r^k. In this iteration, we update every target load to t_r^{k+1}, which is only different from t_r^k for one resource r^* (in our analysis we may disregard iterations in which we do not increase t_{r^*} but decrease $t_{r^*}^{\max}$). Denote the increase in the target loads in iteration j by $n_j = \sum_{r \in R} t_r^{j+1} - t_r^j$ and denote the remaining units to distribute over the resources at the beginning of iteration k by $\bar{n}_k = \mathrm{rk} - \sum_{j=1}^{k-1} n_j$. Denote the strategy profile at the start of this iteration, returned by the oracle in the previous iteration, by $\bar{\mathbf{b}}_k = \bar{\mathbf{b}}_k^1 + \ldots + \bar{\mathbf{b}}_k^n$ (initially $\bar{\mathbf{b}}_0^i = \mathbf{0}$ for all i).

Claim 1. There exists a vector $\hat{\mathbf{b}} = \hat{\mathbf{b}}^1 + \ldots + \hat{\mathbf{b}}^n$ such that $t_r^k \leq \hat{b}_r \leq \max\{b_r^*, t_r^k\}$ for all $r \in R$.

Before proving this claim, we show how the lemma follows from it. The analysis is based on the cost-effectiveness of our greedy choice in the algorithm, e.g. as in [24, Chapter 2]. Consider the quantity $\Delta_k = C(\hat{\mathbf{b}}) - \sum_{r \in R} c_r(t_r^k)$. Using monotonicity of the c_r, we obtain

$$\Delta_k \leq \sum_{r \in R} \left(\max\left\{ c_r(\mathbf{b}^*), c_r(t_r^k) \right\} - c_r(t_r^k) \right) \leq C(\mathbf{b}^*)\,.$$

Note that $(r^*, t_{r^*}^{k+1})$ is a minimizer of the preliminary cost-per-unit in iteration k, and in particular, the sequence of per-unit cost of the load increments in Algorithm 1 is non-decreasing. Also note that Δ_k is the cost of one possible extension from the target loads in iteration k to a strategy profile. Hence, the average cost of one resource in Δ_k is at least the average increase in cost of one resource in iteration k, and we have

$$\frac{\sum_{r\in R} c_r(t_r^{k+1}) - c_r(t_r^k)}{n_k} = \frac{c_{r^*}(t_{r^*}^{k+1}) - c_{r^*}(t_{r^*}^k)}{n_k} \leq \frac{\Delta_k}{\bar{n}_k} \leq \frac{C(\mathbf{b}^*)}{\bar{n}_k} .$$

Using that $\bar{\mathbf{b}}_0 = \mathbf{0}$ and $c_r(0) = 0$ for all $r \in R$, this yields

$$\begin{aligned} C(\mathbf{b}) &= \sum_k \left(\sum_{r\in R} c_r(t_r^{k+1}) - c_r(t_r^k) \right) \leq \sum_k \frac{n_k}{\bar{n}_k} C(\mathbf{b}^*) \\ &= \sum_k \frac{n_k}{\sum_{l\geq k} n_l} C(\mathbf{b}^*) = \sum_k \sum_{j=1}^{n_k} \frac{1}{\sum_{l\geq k} n_l} C(\mathbf{b}^*) \\ &\leq \sum_k \sum_{j=1}^{n_k} \frac{1}{\sum_{l\geq k} n_l - j + 1} C(\mathbf{b}^*) = H_{\mathrm{rk}} C(\mathbf{b}^*) . \end{aligned}$$

It remains to prove Claim 1.

Proof of Claim 1. Set $\hat{\mathbf{b}} = \bar{\mathbf{b}}_k$ and consider a resource r such that $\hat{b}_r > \max\{b_r^*, t_r^k\}$. We call such a resource *overloaded*. Then by Property 1 for $\mathbb{P}_f(\mathrm{rk})$ there exists a resource r' with $b_{r'}^* > \hat{b}_{r'}$ such that $\hat{\mathbf{b}}' = \hat{\mathbf{b}} - \chi_r + \chi_{r'} \in \mathbb{P}_f(\mathrm{rk})$. Replace $\hat{\mathbf{b}}$ by $\hat{\mathbf{b}}'$.

We continue this procedure until there is no overloaded resource anymore. Indeed, as $b_{r'}^* > \hat{b}_{r'}$ we know r' is not overloaded in $\hat{\mathbf{b}}'$. Hence, the total overload $\sum_{r\in R} \max\{\hat{\mathbf{b}}_r - \max\{b_r^*, t_r^k\}, 0\}$ becomes strictly smaller after every replacement. As this quantity is non-negative, this procedure ends and at some point there will not be an overloaded resource anymore. Because it is possible to cover the target loads t_r^k, we know $t_r^k \leq \hat{b}_r \leq \max\{b_r^*, t_r^k\}$ for all $r \in R$. □

Lemma 4. *Algorithm 1 runs in time $\mathcal{O}(m^2\,\mathrm{rk}^2\,Q)$, where Q is the complexity of the value giving oracle that returns $f(U)$ for any $U \subseteq R$.*

Proof. The number of iterations (of the outer while loop) is upper bounded by $m\,\mathrm{rk}$. In every iteration we need to find the minimizer of at most $m\,\mathrm{rk}$ values and then we call the strategy covering oracle, which runs in time $\mathcal{O}(m\,Q)$ by Lemma 2. So the total running time of Algorithm 1 is $\mathcal{O}(m^2\,rk^2\,Q + m^2 rkQ) = \mathcal{O}(m^2\,rk^2\,Q)$. □

This concludes the proof of Theorem 1.

Remark 1. The algorithm is polynomial if we assume that rk is polynomial in the input size, which is the case for special cases such as matroids and graphical polymatroids (arborescences). In particular, for graphical polymatroids, the oracle can be implemented in time $\mathcal{O}(|V|^2\,m\,\log(|V|^2/m))$ [10, Theorem 7.1] (where $|V|$ is the number of vertices in the graph underlying this graphical polymatroid), and we obtain a running time of $\mathcal{O}(m^3\,\mathrm{rk}^2\,|V|^2\log(|V|^2/m))$. However, for general polymatroids, the running time is pseudopolynomial.

Remark 2. As the rank function of a matroid is normalized, monotone and submodular, we obtain a polynomial time H_{rk}-approximation algorithm for matroid congestion games. In these games, players choose subsets rather than multisets of the resources and therefore $\mathrm{rk} \leq n\,m$, implying a polynomial running time. To be more precise, for matroid congestion games, the number of iterations is upper-bounded by $n\,m$ and in every iteration we find the minimizer of at most $n\,m$ values. Using the idea of Cunningham [5], Harks et al [13] proved that their strategy packing oracle can be implemented in time $\mathcal{O}(n^{1.5}\,rkQ)$, where Q is the maximum complexity of the independence

oracles of the matroids[1]. A similar proof also works for our strategy covering oracle and we leave the details to the reader. In particular, the algorithm already provides the decomposition of the strategy profile into player-specific bases $(B_i)_{i \in N}$. The running time of the algorithm for matroid congestion games is thus $\mathcal{O}(n^2\, m^2 + n^{2.5}\, m\, rk Q) = \mathcal{O}(n^{3.5}\, m^2\, Q)$.

However, a comment on the tightness of our approximation guarantee for matroid congestion games is in order. These games are $c \log n$-inapproximable for any $c < 1$ by a reformulation of [15, Theorem 7.1] into our context, but this cannot be strengthened to a $c \log \mathrm{rk}$ hardness result as in Lemma 1 as the construction of game $\mathcal{M}$ from $\mathcal{M}'$ cannot be mimicked within the matroid setting. Thus, as $\log \mathrm{rk} \leq \log n + \log m$, the gap between our approximation guarantee and the hardness result is a constant factor and an additive error of $\log m$. This is a constant gap if $m = poly(n)$. However, if $m \neq poly(n)$, the gap between the inapproximability and the performance guarantee of our algorithm is not constant and the approximation complexity of the problem is yet to be settled exactly.

Acknowledgements. We thank Rico Zenklusen for his contribution to Lemma 2, Rudi Pendavingh and Jorn van der Pol for their contribution to the matroid version of this lemma, and an anonymous reviewer for his helpful comments.

References

[1] H. Ackermann, H. Röglin, and B. Vöcking. On the impact of combinatorial structure on congestion games. *J. ACM*, 55(6):1–22, 2008.

[2] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *J. Comput. System Sci.*, 21(1):136 – 153, 1980.

[3] L. Blumrosen and S. Dobzinski. Welfare maximization in congestion games. *IEEE J. on Sel. Areas in Comm.*, 25(6):1224–1236, 2007.

[4] D. Chakrabarty, A. Mehta, and V. Nagarajan. Fairness and optimality in congestion games. In *Proc. 6th ACM Conf. Electronic Commerce*, pages 52–57, 2005.

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

[6] B. De Keijzer and G. Schäfer. Finding social optima in congestion games with positive externalities. In *Proc. 20th Annual European Sympos. on Algorithms*, pages 395–406, Berlin, Heidelberg, 2012. Springer-Verlag.

[7] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In L. Babai, editor, *Proc. 36th Annual ACM Sympos. Theory Comput.*, pages 604–612, 2004.

[8] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[9] S. Fujishige. *Submodular functions and optimization.* Annals of discrete mathematics. Elsevier, Amsterdam, Boston, Paris, 2005.

[10] H. Gabow. Algorithms for graphic polymatroids and parametric s-sets. *Journal of Algorithms*, 26(1):48–86, 1998.

[11] E. Gassner, J. Hatzl, S. O. Krumke, H. Sperber, and G. J. Woeginger. How hard is it to find extreme Nash equilibria in network congestion games? *Theoret. Comput. Sci.*, 410(47-49):4989–4999, 2009.

[1] The term "maximum complexity" is coined by Cunningham, see [5, Theorem 5.1].

[12] H. Groenevelt. Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *European Journal of Operational Research*, 54(2):227 – 236, 1991.

[13] T. Harks, M. Hoefer, M. Klimm, and A. Skopalik. Computing pure Nash and strong equilibria in bottleneck congestion games. *Math. Program.*, 141(1-2):193–215, 2013.

[14] T. Harks, M. Klimm, and B. Peis. Resource competition on integral polymatroids. In *Web and Internet Economics - 10th International Conference, WINE 2014, Beijing, China, December 14-17, 2014. Proceedings*, pages 189–202, 2014.

[15] T. Harks and P. von Falkenhausen. Optimal cost sharing for capacitated facility location games. *European J. of Oper. Res.*, 239(1):187–198, 2014.

[16] S. Ieong, R. McGrew, E. Nudelman, Y. Shoham, and Q. Sun. Fast and compact: A simple class of congestion games. In *Proc. 20th Natl. Conf. Artificial Intelligence and the 17th Innovative Appl. Artificial Intelligence Conf.*, pages 489–494, 2005.

[17] C. A. Meyers and A. S. Schulz. The complexity of welfare maximization in congestion games. *Networks*, 59(2):252–260, 2012.

[18] I. Milchtaich. Congestion games with player-specific payoff functions. *Games Econom. Behav.*, 13(1):111–124, 1996.

[19] R. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *Internat. J. Game Theory*, 2(1):65–67, 1973.

[20] T. Roughgarden. Barriers to near-optimal equilibria. In *Proc. 55th Annual IEEE Sympos. Foundations Comput. Sci.*, 2014. to appear.

[21] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, Berlin, Germany, 2003.

[22] A. Skopalik and B. Vöcking. Inapproximability of pure Nash equilibria. In *Proc. 40th Annual ACM Sypos. Theory Comput.*, pages 355–364, 2008.

[23] H. Sperber. How to find Nash equilibria with extreme total latency in network congestion games? *Math. Meth. Oper. Res.*, 71(2):245–265, 2010.

[24] V. V. Vazirani. *Approximation Algorithms.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[25] P. von Falkenhausen and T. Harks. Optimal cost sharing for resource selection games. *Math. Oper. Res.*, 38(1):184–208, 2013.

[26] R. Werneck, J. Setubal, and A. da Conceicão. Finding minimum congestion spanning trees. *J. Exp. Algorithmics*, 5, 11, 2000.

[27] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.