Bamas, Étienne ; Lindermayr, Alexander ; Megow, Nicole ; Rohwedder, Lars ; Schlöter, Jens

Santa Claus meets Makespan and Matroids: Algorithms and Reductions

Conference paper	as:	published version (Version of Record)
DOI of this document*(secondary publication):		https://doi.org/10.26092/elib/3186
Publication date of this document:		01/08/2024

* for better findability or for reliable citation

Recommended Citation (primary publication/Version of Record) incl. DOI:

Santa Claus meets Makespan and Matroids: Algorithms and Reductions Étienne Bamas, Alexander Lindermayr, Nicole Megow, Lars Rohwedder, and Jens Schlöter. Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). 2024, pp. 2829-2860. https://doi.org/10.1137/1.9781611977912.

Santa Claus meets Makespan and Matroids: Algorithms and Reductions*

Étienne Bamas† Alexander Lindermayr‡ Nicole Megow‡ Lars Rohwedder§
Jens Schlöter‡

Abstract

In this paper we study the relation of two fundamental problems in scheduling and fair allocation: makespan minimization on unrelated parallel machines and max-min fair allocation, also known as the Santa Claus problem. For both of these problems the best approximation factor is a notorious open question; more precisely, whether there is a better-than-2 approximation for the former problem and whether there is a constant approximation for the latter.

While the two problems are intuitively related and history has shown that techniques can often be transferred between them, no formal reductions are known. We first show that an affirmative answer to the open question for makespan minimization implies the same for the Santa Claus problem by reducing the latter problem to the former. We also prove that for problem instances with only two input values both questions are equivalent.

We then move to a special case called "restricted assignment", which is well studied in both problems. Although our reductions do not maintain the characteristics of this special case, we give a reduction in a slight generalization, where the jobs or resources are assigned to multiple machines or players subject to a matroid constraint and in addition we have only two values. Since for the Santa Claus problem with matroids the two value case is up to constants equivalent to the general case, this draws a similar picture as before: equivalence for two values and the general case of Santa Claus can only be easier than makespan minimization. To complete the picture, we give an algorithm for our new matroid variant of the Santa Claus problem using a non-trivial extension of the local search method from restricted assignment. Thereby we unify, generalize, and improve several previous results. We believe that this matroid generalization may be of independent interest and provide several sample applications.

As corollaries, we obtain a polynomial-time $(2-1/n^{\epsilon})$-approximation for two-value makespan minimization for every $\epsilon > 0$, improving on the previous $(2 - 1/m)$-approximation, and a polynomial-time $(1.75 + \epsilon)$-approximation for makespan minimization in the restricted assignment case with two values, improving the previous best rate of $1 + 2/\sqrt{5} + \epsilon \approx 1.8945$.

*We thank Schloss Dagstuhl for hosting the Seminar 23061 on Scheduling in February 2023 where we had fruitful discussions on this topic.

†Post-Doctoral Fellow, ETH AI Center, Switzerland. etienne.bamas@inf.ethz.ch. This work was partially supported by the Swiss National Science Foundation project entitled "Randomness in Problem Instances and Randomized Algorithms" [project number 200021–184656/1].

‡Faculty of Mathematics and Computer Science, University of Bremen, Germany. {linderal,nmegow,jschloet}@uni-bremen.de.

§Maastricht University, Netherlands. l.rohwedder@maastrichtuniversity.nl. Supported by Dutch Research Council (NWO) project "The Twilight Zone of Efficiency: Optimality of Quasi-Polynomial Time Algorithms" [grant number OCEN.W.21.268].

1 Introduction

In this paper we study two prominent topics from scheduling theory: the SantaClaus problem and unrelated-machine Makespan minimization; in particular, two notoriously difficult questions about polynomial-time approximations that are considered major open problems in the field [6, 28, 32, 33].

In the SantaClaus problem (also known as max-min fair allocation), we are given a set of m players P and a set of n indivisible resources R. Each resource $j \in R$ has unrelated values $v_{ij} \geq 0$ for each player $i \in P$. The task is to find an assignment of resources to players with the objective to maximize the minimum total value assigned to any player. This objective is arguably the best from the perspective of *fairness* for each individual player. The name "Santa Claus" is due to Bansal and Sviridenko [7] who stated this problem as Santa's task to distribute gifts to children in a way that makes the least happy child maximally happy. From the perspective of approximation algorithms, it is entirely plausible that there exists a polynomial-time constant approximation for the problem, with the best lower bound assuming P≠NP being only 2 [8]. On the other hand, the state-of-the-art is "only" a polynomial-time n^{ϵ}-approximation for any constant $\epsilon > 0$, a remarkable result by Chrakabarty, Chuzhoy, and Khanna [9], who also give a polylogarithmic approximation in quasi-polynomial time using the same approach. Positive evidence towards a constant approximation comes from an intensively studied special case, called *restricted assignment* case of SantaClaus, *restricted* SantaClaus in short. Here, the values satisfy $v_{ij} \in \{0, v_j\}$, or equivalently each resource has one fixed value, but can only be assigned to a specific subset of players. The inapproximability from the general case still holds for restricted SantaClaus, even for instances with only two non-zero values, introduced formally later. The first constant approximations have been achieved first by randomized rounding of the so-called configuration LP combined with Lovász Local Lemma (LLL) [7, 16] and later using a sophisticated local search technique analyzed against the configuration LP [2, 3, 12, 13, 15, 19, 26]. The local search method originates from work on hypergraph matchings by Haxell [20].

The "dual" problem with a min-max objective is the equally fundamental and prominent problem of scheduling jobs on unrelated parallel machines so as to minimize the maximum completion time, that is, the makespan. For brevity we refer to this as the Makespan problem. Formally, we are given a set of m machines M and a set of n jobs J. Every job $j \in J$ has size (processing time) $p_{ij} \geq 0$ on machine $i \in M$. The task is to find an assignment of jobs to machines that minimizes the maximum load over all machines. Here, the load of a machine is the total size of jobs assigned to that machine. Lenstra, Shmoys and Tardos [25] gave a beautiful 2-approximation algorithm based on rounding a sparse vertex solution of the so-called assignment LP (a simpler relaxation than the configuration LP). The rounding has been slightly improved to the factor $2 - 1/m$ [29], but despite substantial research efforts, this upper bound remains undefeated. The best lower bound on the approximability assuming P≠NP is 3/2 [25]. Similar to the SantaClaus problem, the restricted assignment case with $p_{ij} \in \{p_j, \infty\}$ has also been studied extensively for the Makespan problem, referred to as *restricted* Makespan. However, the barrier of 2 has been overcome only partially even in this setting: with non-constructive integrality gap bounds [31] and better-than-2 approximations in quasi-polynomial time [22] and for the special case of two sizes [1, 10].

Intuitively, the two problems are related and in the community the belief has been mentioned that SantaClaus admits a constant approximation if (and only if) Makespan admits a better-than-2 approximation; see e.g., [5, 6]. Indeed, techniques for one problem often seem to apply to the other one, but no formal reductions are known. We give some examples for such parallels:

(*i*) The configuration LP, see [7], is the basis of all mentioned results for the restricted assignment variant of both problems.

(*ii*) The local search technique by Haxell [20] for hypergraph matching has been adopted and shown to be very powerful for both problems. First, it has been picked up for restricted SantaClaus [2, 3, 15, 26] and later it has been further developed for restricted Makespan [1, 22, 31].

(*iii*) Chakrabarty, Khanna and Li [10] transferred the technique of rounding the configuration LP via LLL used for restricted SantaClaus [16, 18] to restricted Makespan with two job sizes and thereby provided the first slightly-better-than-two approximation in polynomial time.

(*iv*) The reduction for establishing hardness of approximation less than 2 for SantaClaus [8] is essentially the same as the earlier construction for the 3/2-inapproximability for Makespan [25].

(*v*) The LP rounding by Lenstra et al. [25] achieves an additive approximation within the maximum (finite) processing time $p_{\max}$, i.e., the makespan is at most $\text{OPT} + p_{\max}$, which can be translated into a multiplicative 2-approximation. Bezakova and Dani [8] show the same additive approximation for SantaClaus: each player is guaranteed a value at least $\text{OPT} - v_{\max}$. Note that for the max-min objective this does not

translate into a multiplicative guarantee.

In this paper, we confirm (part of) the conjectured relation between the MAKESPAN and the SANTACLAUS problem with respect to their approximability. As our first main result we prove that a better-than-2 approximation for MAKESPAN implies an $\mathcal{O}(1)$-approximation for SANTACLAUS.

THEOREM 1.1. *For any $\alpha \geq 2$, if there exists a polynomial-time $(2-1/\alpha)$-approximation for* MAKESPAN*, then there exists a polynomial-time $(\alpha+\epsilon)$-approximation for* SANTACLAUS *for any $\epsilon > 0$.*

For values of $\alpha < 2$, a $(2-1/\alpha)$-approximation for the MAKESPAN problem is NP-complete and the implication would still hold, even though clearly uninteresting. Similarly to this, we will restrict our attention in later theorems to non-trivial values of α.

We prove also the reverse direction for the *two-value* case, in which all resource values are $v_{ij} \in \{0, u, w\}$, for some $u, w \geq 0$, and all processing times are $p_{ij} \in \{u, w, \infty\}$, for some $u, w \geq 0$, respectively. This implies the equivalence of MAKESPAN and SANTACLAUS in that case.

THEOREM 1.2. *For any $\alpha \geq 2$, there exists an α-approximation algorithm for two-value* SANTACLAUS *if and only if there exists a $(2-1/\alpha)$-approximation algorithm for two-value* MAKESPAN*.*

We then move to the restricted assignment case, where one might hope to unify previous results and possibly infer new results by showing a similar relationship. Using our techniques, however, this seems unclear, since the aforementioned reductions do not maintain the characteristics of the restricted assignment case. However, it turns out to be useful to consider a matroid generalization of our problems. Towards this, we briefly introduce the notion of a matroid. A *matroid* is a non-empty, downward-closed set system $(E, \mathcal{I})$ with ground set E and a family of subsets $\mathcal{I} \subseteq 2^E$, which satisfies the *augmentation property*:

$$\text{if } I, J \in \mathcal{I} \text{ and } |I| < |J|, \text{ then } I + j \in \mathcal{I} \text{ for some } j \in J \setminus I. \tag{1.1}$$

Given a matroid $\mathcal{M} = (E, \mathcal{I})$, a set $I \subseteq E$ is called *independent* if $I \in \mathcal{I}$, and *dependent* otherwise. An inclusion-wise maximal independent subset is called a *basis* of $\mathcal{M}$, and we denote the set of bases by $\mathcal{B}(\mathcal{M})$. With a matroid $\mathcal{M}$, we associate a rank function $r : 2^E \to \mathbb{Z}_{\geq 0}$, where $r(X)$ describes the maximal cardinality of an independent subset of X. Typical examples of matroids include: linearly independent subsets of some given vectors, acyclic edge sets of a given undirected graph, and subsets of cardinality bounded by some given value. A *polymatroid* is the multiset analogue of a matroid. We refer to Section 1.3 for further definitions and to [27] for a general introduction to matroids.

Moving back to our two problems, we first introduce the restricted *resource-matroid* SANTACLAUS problem, where we consider again an input of resources and players and each resource j has a value v_j for each player i as in the restricted assignment case. In the matroid variant, however, each resource can potentially be assigned to multiple players, subject to a (poly-)matroid constraint; more precisely, we require the set of players, which we assign the resource to, to be a basis of a given resource-specific (poly-)matroid, and the resource contributes to the total value of each of these players. As one may also consider a more general variant with unrelated values v_{ij} we use the phrase *restricted* to emphasize our restricted assignment model. Similarly, we define a restricted *job-matroid* MAKESPAN problem by replacing the max-min with a min-max objective and asking for an assignment of each job to a set of machines which forms a basis in the job's matroid.

Davies, Rothvoss and Zhang [15] recently introduced a closely related matroid variant of restricted SANTACLAUS. Their variant, however, is significantly more restrictive and can be summarized as follows: they allow a single infinite value matroid-resource and a set of "small value" traditional resources (without the matroid generalization). For this variant they give a $(4+\epsilon)$-approximation algorithm.

First, we show that in our general variant an analogous relation to Theorem 1.2 holds.

THEOREM 1.3. *For any $\alpha \geq 2$, there exists a polynomial-time α-approximation algorithm for the restricted two-value resource-matroid* SANTACLAUS *problem if and only if there exists a polynomial-time $(2-1/\alpha)$-approximation algorithm for the restricted two-value job-matroid* MAKESPAN *problem.*

Since the matroid version is a new problem, we cannot directly infer any approximation results for it. Hence, we develop a polynomial-time approximation algorithm for the restricted resource-matroid SANTACLAUS problem.

THEOREM 1.4. *For any $\epsilon > 0$, there exists a polynomial-time $(8+\epsilon)$-approximation algorithm for the restricted resource-matroid* SANTACLAUS *problem and a $(4+\epsilon)$-approximation in the two-value case.*

This is achieved via a non-trivial generalization of the commonly used local search technique for restricted assignment, see Section 1.2 for a technical overview. We prove the variant for two values and then give a reduction from the general case, see Lemma 3.1. Apart from the curiosity-driven motivation for a matroid generalization of the classical scheduling problems and the usage through our reductions, we present in Section 1.1 sample applications where such a variant arises.

We note that some special cases of the job-matroid MAKESPAN problem have already been considered in the past. Indeed, Azar et al. [4] give a 2-approximation for the MAKESPAN problem where each job j needs to be processed by k_j different machines, which is the special case where each job is equipped with a uniform matroid[1]. Here, the machines are considered unrelated hence the processing times p_{ij} can be arbitrary. We give a generalization of the result by Azar et al. which allows an arbitrary matroid for each job.

THEOREM 1.5. *There exists a polynomial-time algorithm for the unrelated job-matroid* MAKESPAN *problem which returns a solution of value at most $\min\{2\mathrm{OPT}, \mathrm{OPT} + p_{\max}\}$, where $p_{\max} = \max_{i\in M, j\in J, p_{ij}<\infty} p_{ij}$ is the biggest bounded processing time.*

The proof of this result can be found in Appendix C. It is based on a non-trivial generalization of a rounding result by Shmoys and Tardos [30]. Interestingly, Azar et al. [4] mention that the rounding theorem of Lenstra, Shmoys, Tardos [25] cannot be applied in the matroid setting, because it crucially relies on a counting argument which does not hold anymore. Fortunately, the result in [30] does not need this argument. As in the simple job case, we obtain a similar theorem for the resource-matroid SANTACLAUS problem.

THEOREM 1.6. *There exists a polynomial-time algorithm for the unrelated resource-matroid* SANTACLAUS *problem which returns a solution of value at least $\mathrm{OPT} - v_{\max}$, where $v_{\max} = \max_{i\in M, j\in J} v_{ij}$ is the biggest resource value.*

The proof of this last theorem can easily be obtained from a rounding theorem proved in Section 3 (Theorem 3.1).

To conclude this section, we state two immediate implications of our results to the state-of-the-art of the MAKESPAN problem.

COROLLARY 1.1. *For every $\epsilon > 0$, there exists a polynomial-time $(1.75+\epsilon)$-approximation algorithm for the restricted two-value job-matroid* MAKESPAN *problem.*

Corollary 1.1 holds in particular true for the restricted MAKESPAN problem, thus improving upon the previously best polynomial-time approximation rate of $1 + 2/\sqrt{5} + \epsilon \approx 1.8945$ [1]. The corollary follows from combining Theorems 1.3 and 1.4. In their work on restricted SANTACLAUS, Davies et al. [15] managed to reduce the technical complexity of previous works, which handled complicated path decompositions explicitly, using a cleaner matroid abstraction. Our algorithm shows that such a simplification is also possible for restricted two-value MAKESPAN, which was not clear before.

COROLLARY 1.2. *For every $\epsilon > 0$, there exists a polynomial-time $(2 - 1/n^{\epsilon})$-approximation algorithm and a quasi-polynomial-time $(2 - 1/\mathrm{polylog}(n))$-approximation algorithm for two-value* MAKESPAN.

This result follows from the algorithm of Chakrabarty et al. [9] and Theorem 1.2, and improves upon the best-known polynomial-time approximation factor of $2 - 1/m$ for m machines [29].

1.1 Applications The job-matroid MAKESPAN problem already has some known applications in data retrieval mentioned in [4]. To complement this, we lay out three additional applications of our matroid generalization.

First, consider service centers that offer various types of services to clients. The specific service that such a center offers has some value associated with it and it can only be provided to a limited number of clients, a typical constraint appearing for example in capacitated facility location problems. Furthermore, a service center can serve only clients that are located in the same region and a client can only receive a specific type of service

[1] A *uniform matroid* $\mathcal{M} = (X, \mathcal{I})$ of rank r has as independent sets all subsets of X of cardinality at most r.

once, i.e., by a single center, since receiving the same service twice yields no additional value. The services should be assigned to the clients in such a way that all clients are treated "fairly" with respect to their total value for the services. That is, we want to maximize the total value for the least happy client. This can be modeled as a resource-matroid SantaClaus problem with clients being players and services (one per service type) being resources. The set of clients that can receive a particular type of service can be modeled as a transversal matroid[2]. In the classical (restricted) SantaClaus problem, one cannot express the constraint that a client can receive each type of service only once.

As a second example, consider a program committee (PC) for a scientific conference. We would like to assign papers to PC members such that the workload is balanced in the sense that we minimize the maximum workload over all PC members. We will view this as a Makespan problem with PC members being machines and submissions being jobs. PC members have declared which submissions they would agree to assess. Submissions may be of different types such as "short papers" or "regular paper" with varying workloads. In a typical conference, each submission needs to be assessed three times and obviously it is important that this is done by different PC members; hence, we cannot simply model this as a traditional restricted assignment problem where we duplicate a job three times. However, it is easy to model this using matroids by having one basis for each triple of PC members that agree to assess this submission, i.e., we have a job-matroid Makespan problem with a uniform matroid of rank 3.

Our third illustration is a job-matroid Makespan problem, in which a graphic matroid[3] allows us to model connectivity requirements. In cloud computing and data centers, a number of servers is available to execute multiple applications at the same time. Each application is executed on a subset of servers and these servers must be connected to allow for communication. We assume that these connections are direct in the sense that an application may not use additional servers as Steiner nodes. We need to reserve a certain bandwidth for each application's communication, which depends on characteristics of the application itself. The task is to select carefully on which links to reserve the bandwidth for each individual application such that load on these links is balanced, more precisely, we want to select links to minimize the maximum total bandwidth requirement imposed on any link. This can be modelled as job-matroid Makespan problem with jobs being applications, machines being the edges (links) in a graph formed by the allocated servers, and the load (processing time) being the requested bandwidth of the application. The task is to choose for each job a spanning tree, that is, a basis in the job-dependent graphic matroid such that the maximum total bandwidth on any edge is minimized. This cannot be modeled as classical Makespan or restricted assignment problem, since it cannot capture the structure of a graphic matroid.

1.2 Algorithmic techniques Our main algorithmic contribution lies in a local search algorithm for the new restricted resource-matroid SantaClaus problem, see Theorem 1.4. We give an overview of the method here, its main technical merits, and how it relates to previous works. The specific local search method that we refer to originates in an algorithmic proof for a hypergraph matching theorem by Haxell [20]. The theorem is a generalization of Hall's theorem for bipartite graphs to hypergraphs and Haxell's proof can be thought of as a very non-trivial extension of the augmenting path method in bipartite graphs. Asadpour et al. [3] made the connection to restricted SantaClaus. In addition to a black-box reduction to the specific hypergraph matching problem, they also reinterpreted Haxell's method as a sophisticated algorithm for restricted SantaClaus.

Although not explicitly mentioned in earlier works, this new algorithm can be thought of as a generalization of the typical augmentation algorithm for matroid partition: the core of the problem lies in the case where we have two values for the resources, more precisely, either the value of a resource is infinitely large or it is a unit value 1. This case is up to constants equivalent to the general problem, see e.g., [7]. Observe now the following structure: we need to select a subset I_M of players such that there exists a left-perfect matching of I_M to the infinite-value resources and such that there exists a b-matching of all players in $P \setminus I_M$ to the small resources (each player is matched to b resources, each resource to at most one player), where b has to be maximized. The sets I_M that fulfill the condition above form a transversal matroid, but unfortunately the sets of players for which there is a b-matching does not (for a fixed $b > 1$). If they *would* actually form a matroid, then the problem could easily

[2]Given a bipartite graph $G = (J \cup S, E)$, a set $S' \subseteq S$ is independent in the *transversal matroid* $\mathcal{M} = (S, \mathcal{I})$ if there is a matching in G which covers S'.

[3]Given an undirected graph $G = (V, E)$, the *graphic matroid* $\mathcal{M} = (E, \mathcal{I})$ has as independent sets the cycle-free edge sets (forests), i.e., $\mathcal{I} = \{F \subseteq E : F \text{ is acyclic in } G\}$.

be solved by matroid partition, where given two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1), \mathcal{M}_2 = (E, \mathcal{I}_2)$ over the same ground set, we want to find two independent sets $I_1 \in \mathcal{I}_1$, $I_2 \in \mathcal{I}_2$ that partition the ground set, i.e., $I_1 \dot\cup I_2 = E$ (here we focus on the variant with two matroids, although also more than two matroids may be allowed). Matroid partition–and the equivalent problem of matroid intersection–can be solved in polynomial time by an augmenting path algorithm that repeatedly increases the union of I_1 and I_2 by first swapping elements between these two sets. Although, as mentioned above, the b-matching does not have a matroid structure, the algorithmic paradigm of swapping elements between matching and b-matching in order to increase their union still works once we allow approximation of b.

The constraint on set I_M forms a transversal matroid (implied by the infinite-value resources) and Davies et al. [15] then showed that the algorithmic idea generalizes to arbitrary matroid structures. In their problem, however, the b-matching remains the same without further abstraction, whereas in our further generalization, we embrace the polymatroid structure of the b-matching. We now require instead of a b-matching that the multiset $b \cdot (P \setminus I_M)$ (having b copies for each element in $P \setminus I_M$) is in some given polymatroid. We believe that this abstraction is the logical conclusion for this line of research.

Although a seemingly natural extension, it is highly non-trivial to generalize the existing algorithm to our setting. Firstly, there are conceptional issues that come from the fact that previous methods revolve around reassigning resources or jobs and those do not exist explicitly in our polymatroid. Secondly, a serious technical problem comes from the lack of a certificate of infeasibility. The design of the algorithm is closely connected to a certificate of infeasibility, which is provided (for analysis' sake) in case the algorithm fails. For example, in matroid partition when the augmenting path algorithm fails, one can derive a set X such that $r_1(X)+r_2(X) < |X|$, where r_1 and r_2 are the rank functions of the matroids [24]. This clearly proves infeasibility. In applications of the method to restricted SantaClaus or Makespan, as a certificate of infeasibility a dual solution to the configuration LP is usually constructed. However, neither of the two ideas generalize to our setting, as we will lay out in Section 5. Hence, we come up with a novel certificate, which may be of independent interest.

1.3 Definitions and notation We write $\mathcal{O}_\epsilon(\cdot)$ as the standard $\mathcal{O}$-notation, where we suppress any factors that are functions in only ϵ. For a set X and an element i we write $X + i := X \cup \{i\}$. Similarly, $X - i := X \setminus \{i\}$.

Let E be a universe. For a vector $x \in \mathbb{R}^E$, we write $x(e)$ for the entry of x corresponding to $e \in E$, and $x(S) = \sum_{e \in S} x(e)$. For some $X \subseteq E$, we write $b \cdot X$ as the vector $y \in \mathbb{Z}^E$ with $y(e) = b$ for $e \in X$ and $y(e) = 0$ for $e \notin X$. A set function $f : 2^E \to \mathbb{R}$ is submodular if for all subsets $A, B \subseteq E$ holds $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, and monotone if for all $A \subseteq B \subseteq E$ holds $f(A) \leq f(B)$. Let $f : 2^E \to \mathbb{Z}_{\geq 0}$ be a monotone submodular integer function with $f(\emptyset) = 0$. An *integer polymatroid* over E associated with f is defined as

$$\mathcal{P} = \{x \in \mathbb{Z}^E_{\geq 0} : x(S) \leq f(S)\ \forall S \subseteq E\}.$$

In the following we always refer to integer polymatroids when talking about polymatroids. A polymatroid can be interpreted as the multiset generalization of a matroid and most concepts of matroids translate easily to polymatroids. Every element $x \in \mathcal{P}$ can be seen as an independent multiset in which an element $e \in E$ appears with multiplicity $x(e)$. A polymatroid is also downward-closed, that is, $x \in \mathcal{P}$ implies $y \in \mathcal{P}$ for any $0 \leq y \leq x$, and satisfies the augmentation property, that is, if $x, y \in \mathcal{P}$ with $x(E) < y(E)$, then there is some $e \in E$ such that $x' \in \mathcal{P}$ with $x'(e) = x(e) + 1$ and $x'(e') = x(e')$ for all $e' \in E - e$. In particular, any matroid is a polymatroid.

A basis of a polymatroid $\mathcal{P}$ is an element $x \in \mathcal{P}$ which satisfies $f(E) = x(E)$, meaning that all bases have the same cardinality (in terms of multisets). We denote the set of bases of $\mathcal{P}$ by $\mathcal{B}(\mathcal{P})$. For a given polymatroid $\mathcal{P}$ and a constant $k \in \mathbb{Z}_{\geq 0}$, the set $\{x \in \mathcal{P} : x(e) \leq k\ \forall e \in E\}$ is again a polymatroid.

For a given polymatroid $\mathcal{P}$ of the submodular function f, and some vector $z \in \mathbb{Z}^E_{\geq 0}$ with $x \leq z$ for all $x \in \mathcal{P}$, we define the *dual polymatroid* $\overline{\mathcal{P}}$ of $\mathcal{P}$ with respect to z via the set function g with

$$g(S) = z(S) + f(E \setminus S) - f(E)$$

for every $S \subseteq E$. This function is submodular, monotone and satisfies $g(\emptyset) = 0$, hence this definition is well-defined. Note that if $x \in \mathcal{B}(\mathcal{P})$ it follows $g(E) = z(E) + f(\emptyset) - f(E) = z(E) - x(E)$, and therefore $z - x \in \mathcal{B}(\overline{\mathcal{P}})$.

If a polymatroid $\mathcal{P}$ associated with a function f is given as an input for a problem, we assume that it is represented in form of a value oracle for f. We can test whether some vector x is in $\mathcal{P}$ by checking whether the minimum of the submodular function $f(S) - x(S)$ is non-negative, which can be done with a polynomial number of value queries to f. We refer for an extensive overview over polymatroids to Schrijver [27, chapters 44 - 49].

We now give precise definitions of the matroid problems we consider.

DEFINITION 1.1. (RESOURCE-MATROID SANTACLAUS) *In the restricted* resource-matroid SANTACLAUS *problem, there are sets of m players P and n resources R with values v_j for all $j \in R$. Further, for every resource $j \in R$ there is an integer polymatroid $\mathcal{P}_j$ over P. The task is to allocate each resource $j \in R$ to a basis $x_j \in \mathcal{B}(\mathcal{P}_j)$ and let each player i profit from the resource j with value $v_j \cdot x_j(i)$. The goal is to maximize the minimum total value any player receives, i.e.,* $\min_{i \in P} \sum_{j \in R} v_j \cdot x_j(i)$.

DEFINITION 1.2. (JOB-MATROID MAKESPAN) *In the restricted* job-matroid MAKESPAN *problem, there are sets of m machines M and n jobs J with sizes p_j for all $j \in J$. Further, for every job $j \in J$ there is an integer polymatroid $\mathcal{P}_j$ over M. The task is to allocate each job $j \in J$ to a basis $x_j \in \mathcal{B}(\mathcal{P}_j)$ which means that j contributes load $p_j \cdot x_j(i)$ to the total load of machine i. The goal is to minimize the maximum total load over all machines, i.e.,* $\max_{i \in M} \sum_{j \in J} p_j \cdot x_j(i)$.

These new matroid allocation problems generalize the restricted assignment variants of SANTACLAUS and MAKESPAN, respectively. In fact, the matroid variant with a uniform matroid of rank 1 corresponds to the respective traditional problem.

Note that in restricted resource-matroid SANTACLAUS it is equivalent to require that $x_j \in \mathcal{P}_j$ for each resource j instead of $x_j \in \mathcal{B}(\mathcal{P}_j)$, since we can always increase x_j to reach a basis without making the solution worse. In restricted job-matroid MAKESPAN this is not the case.

Both matroid problems can be reduced to instances where the number of polymatroids is equal to the number of distinct job sizes (resource values). This is because we can sum polymatroids associated with jobs (resources) of equal size (value) to a single one, and then decompose a basis for such a merged polymatroid into bases for the original polymatroids via polymatroid intersection. Formally, we get the following proposition. For a more detailed explanation, see Appendix A.

PROPOSITION 1.1. *For any $\alpha \geq 1$, if there exists a polynomial-time α-approximation algorithm for restricted job-matroid* MAKESPAN *(resource-matroid* SANTACLAUS*) with h jobs (resources), then there exists a polynomial-time α-approximation algorithm for restricted job-matroid* MAKESPAN *(resource-matroid* SANTACLAUS*) with p_j resp. $v_j \in \{w_1, \ldots, w_h\}$ and $w_1, \ldots, w_h \geq 0$.*

2 Santa Claus and makespan reductions

In this section we present our first two reductions and prove Theorem 1.1 and Theorem 1.2. The precise statements given in the following subsections imply these results. They are formulated as subroutines for a standard guessing framework (see e.g. [21]), which we briefly explain here. Consider a SANTACLAUS instance I for which we want to compute an α-approximate solution. We first guess $\mathrm{OPT}(I)$ with some variable T using binary search as follows. For some guess T, we scale down all values of I by factor T and obtain I'. Then, we prove that if $\mathrm{OPT}(I) \geq T$ (and $\mathrm{OPT}(I') \geq 1$), our subroutine finds a solution for I' with an objective value of at least $1/\alpha$. If we do not obtain such a solution, we can conclude $\mathrm{OPT}(I) < T$ and safely repeat with a smaller guess. Otherwise, we repeat with a larger guess. After establishing $T = \mathrm{OPT}(I)$, the subroutine gives us a solution with an objective value of at least T/α. For MAKESPAN one can design an analogous procedure.

2.1 From Santa Claus to makespan minimization

In this section, we present an approximation preserving reduction (up to a factor of $1+\epsilon$) from SANTACLAUS to MAKESPAN. More precisely, we show the following result.

LEMMA 2.1. *For any $\alpha \geq 2$ and $\epsilon > 0$, given an instance I of* SANTACLAUS *with $\mathrm{OPT}(I) \geq 1$, we can construct in polynomial time an instance I' of* MAKESPAN *such that, given a $(2-1/\alpha)$-approximate solution for I', we can compute in polynomial time a solution for I with an objective value of at least $1/(\alpha+\epsilon)$.*

This lemma then implies Theorem 1.1 via the guessing framework. We split the proof of Theorem 2.1 into two parts (cf. Lemmas 2.2 and 2.3). First, we show that we can define a polynomial number of *configurations* for each player, which represent different options this player has. We show that there is a nearly optimal solution, up to a factor of $1+\epsilon$, that only uses these configurations.

Second, we reduce this problem to MAKESPAN without losing additional constants. That is, we present a reduction proving that a $(2-1/\alpha)$-approximation for MAKESPAN implies an α-approximation for SANTACLAUS

restricted to polynomially many configurations. Intuitively, the fact that we can enumerate the list of possible optimal configurations per player enables us to create gadgets for every configuration in the constructed MAKESPAN instance which we exploit when reducing solutions.

Santa Claus with polynomially many configurations. Consider a SANTACLAUS instance with players P and n resources R. We define the set of *value types* as $\mathcal{T} = \{v_{ij} : i \in P, j \in R\}$, which contains all distinct resource values that occur in the instance. We call a function $c : \mathcal{T} \to \{0, 1, \dots, n\}$ a *configuration*, and define the *total value* of c as $|c| = \sum_{v \in \mathcal{T}} c(v) \cdot v$. One can also see a configuration as a multiset of value types.

Given a configuration c_i for a player i of a SANTACLAUS instance I, we say that a resource assignment $A = \{A_i\}_{i \in P}$ for I that assigns the set of resources A_i to player i *matches* the configuration c_i if $|\{j \in A_i : v_{ij} = v\}| = c_i(v)$ for every value type $v \in \mathcal{T}$.

We use $\mathcal{C}_i$ to refer to a set of configurations for a player $i \in P$ and call $\mathcal{C} = \{\mathcal{C}_i\}_{i \in P}$ a *collection of configurations*. A resource assignment A matches a collection of configurations $\mathcal{C}$ if, for each player i, there exists a configuration $c \in \mathcal{C}_i$ such that A_i matches c. Given a SANTACLAUS instance I and a collection of configurations $\mathcal{C} = \{\mathcal{C}_i\}_{i \in P}$, we use $\mathrm{OPT}_{\mathcal{C}}(I)$ to refer to the optimal objective value for instance I among those solutions that match $\mathcal{C}$.

The main result of this section is the following lemma.

LEMMA 2.2. *For every $\epsilon > 0$ and a given instance I of* SANTACLAUS *with $\mathrm{OPT}(I) \geq 1$, we can construct a rounded instance I' with a collection of configurations $\mathcal{C}$ such that the number of configurations for each player is polynomial in the input size of I and $\mathrm{OPT}_{\mathcal{C}}(I') \geq 1/(1+\epsilon)$.*

Further, every solution for I' of objective value T is a solution for I with objective value at least T.

The lemma essentially allows us to consider only solutions that partially match the constructed collection of configurations $\mathcal{C}$. If we find such a solution that α-approximates $\mathrm{OPT}_{\mathcal{C}}(I')$, we immediately get a $(\alpha + \epsilon)$-approximation for $\mathrm{OPT}(I)$.

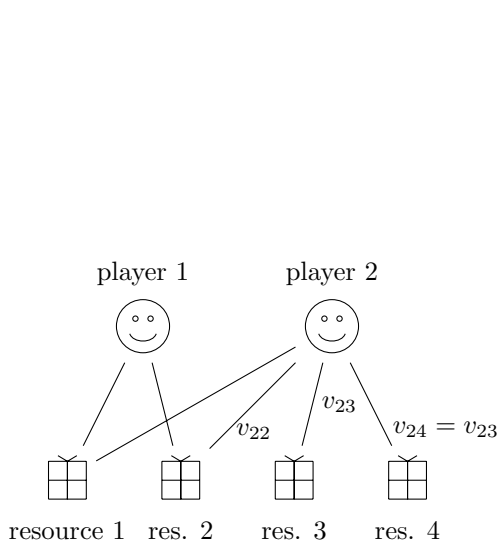
To prove the lemma, we employ several rounding techniques and enforce a certain monotonicity condition on the configurations. This allows us to reduce the number of configurations per player to a polynomial while still guaranteeing $\mathrm{OPT}_{\mathcal{C}}(I') \geq 1/(1+\epsilon)$. For a full proof, we refer to Appendix B.

Reduction to makespan minimization. We prove the following lemma which, together with Lemma 2.2, implies Lemma 2.1.

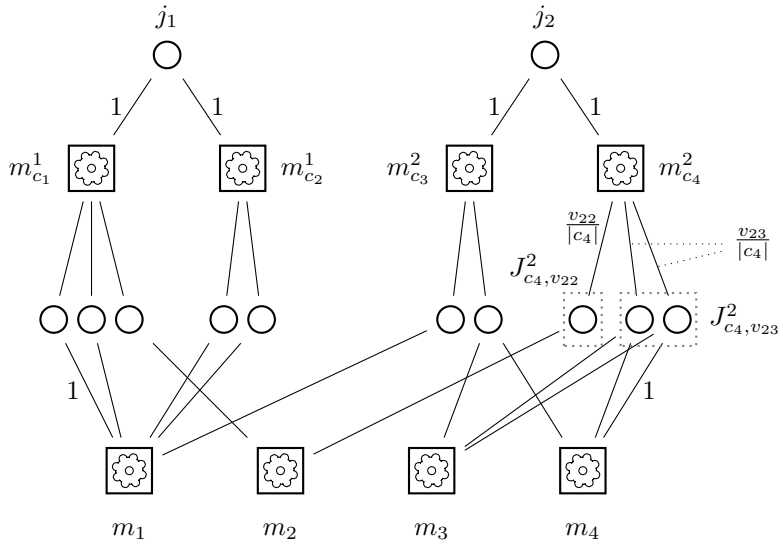
LEMMA 2.3. *Let I be an instance of* SANTACLAUS *and let $\mathcal{C}$ be a collection of configurations with $\mathrm{OPT}_{\mathcal{C}}(I) \geq 1$. For any $\alpha \geq 1$, we can construct in polynomial time an instance I' of* MAKESPAN *such that, given a $(2 - 1/\alpha)$-approximate solution for I', we can compute in polynomial time a solution for I with value at least $1/\alpha$. The running times are polynomial in the size of $(I, \mathcal{C})$.*

We first give some intuition for the reduction of the lemma. Assume for simplicity that every configuration in $\mathcal{C}$ has value 1. We want to construct a MAKESPAN instance with an optimal makespan equal to 1. Exploiting the polynomial number of configurations, we introduce *configuration-machines* for every player and every configuration of that player. By using a gadget structure, we ensure that every solution for the MAKESPAN instance "selects" one configuration-machine for each player, which we call the *player-machine*. This is done by forcing an extra (selection-)load of 1 on exactly one configuration-machine for that player. The other configuration-machines will just be able to absorb all other jobs that can be placed on the machine, so we assume that they do so and ignore them. Intuitively, the player-machine determines the configuration which we (partially) use for the player when transferring a solution back to the SANTACLAUS instance. To this end, we encode the corresponding configuration c of the player-machine by introducing for every $v \in \mathcal{T}$ a total of $c(v)$ *configuration-jobs* with size v on that machine. For every resource we also introduce a *resource-machine*, on which only configuration-jobs of the same value type as the corresponding resource can be placed, with size 1. In an optimal solution of makespan 1, no configuration-job can be placed on a player-machine due to the selection-load. This means that all these jobs have to be placed on the resource-machines instead. Since each resource-machine can absorb at most one job, we can interpret the placement of the configuration jobs as a resource assignment for the SANTACLAUS instance that matches the configuration of the player-machines. This, however, only works for optimal solutions.

To better understand the connection between approximate solutions, imagine an initial state where all configuration-jobs are placed on their player-machines in the MAKESPAN instance, and all resources are unassigned in the SANTACLAUS instance. This means that all player-machines have a total load of $1 + |c| = 2$ (observe that



(a) SantaClaus instance I. Players are visualized by smileys and resources by gifts.



(b) Makespan instance I'. Machines are visualized by squares with a gear and jobs by circles.

Figure 1: The construction used in Lemma 2.3. In both pictures an edge indicates that an item has a non-trivial value for an entity. Note that $v_{23} = v_{24}$ and thus resources 3 and 4 belong to the same value class of player 2. Thus, the configuration-jobs in $J^2_{c_4,v_{23}}$ have edges to both m_3 and m_4. The given configuration collections are $\mathcal{C}_1 = \{c_1, c_2\}$ and $\mathcal{C}_2 = \{c_3, c_4\}$, and the support of configuration c_4 is given by $c_4(v_{23}) = 2$ and $c_4(v_{22}) = 1$.

$|c| = 1$ is caused by the configuration-jobs) and all players have a total value of 0. Now, a player can gain a resource by moving a suitable configuration-job away from her player-machine to a resource-machine. Since, even in a better-than-2 approximation for the Makespan instance, a resource-machine can absorb at most one job, we can again interpret this as the players competing for the resources via moving jobs away from their player-machines to resource-machines. Therefore, in a $(2-1/\alpha)$-approximation for the Makespan instance, a player must be able to move jobs away from her player-machine of total size at least $1/\alpha$. But this means in our interpretation that she receives resources of total value at least $1/\alpha$ in the SantaClaus instance.

In the following, we formalize these ideas and prove Lemma 2.3. Fix an instance I of the SantaClaus problem and a collection of configurations $\mathcal{C}$ for I with $\mathrm{OPT}_{\mathcal{C}}(I) \geq 1$. We proceed by constructing the Makespan instance I' as follows:

1. Remove all configurations from $\mathcal{C}$ of value strictly less than 1.
2. For every player i introduce a player-job j_i in I'.
3. For every resource j introduce a resource-machine m_j in I'.
4. For every player i and every configuration $c \in \mathcal{C}_i$ introduce a configuration-machine m^i_c in I', where the size of the player-job j_i is equal to 1 on every configuration-machine m^i_c and ∞ on all other machines.
5. For every player i, every configuration $c \in \mathcal{C}_i$ and every value type $v \in \mathcal{T}$ introduce a set $J^i_{c,v}$ of $c(v)$ many configuration-jobs. A configuration-job in $J^i_{c,v}$ has size 1 on every resource-machine m_j if resource j has value type v for player i (i.e., $v_{ij} = v$), size $v/|c|$ on the configuration-machine m^i_c, and ∞ on all other machines.

Since we assumed that $\mathrm{OPT}_{\mathcal{C}}(I) \geq 1$, the first step does not affect $\mathrm{OPT}_{\mathcal{C}}(I)$. See Figure 1 for an example of this reduction. We prove two auxiliary lemmas that imply Lemma 2.3.

Lemma 2.4. *The optimal objective value of I' is at most* 1.

Proof. Fix a solution of I that is optimal among the solutions that match $\mathcal{C}$. Consider a player i of instance I and let $c \in \mathcal{C}_i$ be the selected configuration for player i in the given solution. Let A_i be the set of resources assigned to player i. In the solution for I', we assign job j_i to machine m_c^i, giving it a load of 1. Further, we assign the configuration-jobs $J_{c,v}^i$ of configuration c to resource-machines $\{m_j : j \in A_i\}$ such that every resource-machine receives at most one job. Such an assignment must exist by the fact that configuration c is matched by the fixed solution for I. For every configuration $c' \in \mathcal{C}_i \setminus \{c\}$ we assign for all $v \in \mathcal{T}$ every configuration-job in $J_{c',v}^i$ to machine $m_{c'}^i$, giving those a load of $\sum_{v \in \mathcal{T}} c'(v)\frac{v}{|c'|} = 1$. Since, in the given solution, every resource j is assigned to at most one player i, and since we have assigned at most one configuration-job to machine m_j, every resource-machine also has a load of at most 1. Hence, the makespan of the constructed solution for I' is at most 1. ∎

LEMMA 2.5. *For any $\alpha \geq 1$, given a solution for I' with a makespan of at most $2 - 1/\alpha$, we can construct in polynomial time a solution for I where every player receives a total value of at least $1/\alpha$.*

Proof. Given a solution for I' where every machine has a load of at most $2 - 1/\alpha$, we construct a solution for I as follows. Fix a player i and assume that j_i is assigned to machine m_c^i.

Let J_i be the set of configuration-jobs of configuration c of player i which are *not* assigned to m_c^i. Thus, every job in J_i is assigned to a resource-machine. Note that every resource-machine has at most one assigned job, because every job has size of at least 1 on these machines. Let R_i be the set of resources for which the corresponding resource-machines receive a job of J_i. We assign the resources R_i to player i in the solution for I. The load contributed by configuration-jobs to machine m_c^i is at most $1 - 1/\alpha$, because job j_i is also assigned to m_c^i and has size 1. This implies that the total size of jobs in J_i for machine m_c^i is at least

$$\sum_{j \in J_i : j \in J_{c,v}^i} \frac{v}{|c|} \geq \left(\sum_{v \in \mathcal{T}} c(v)\frac{v}{|c|}\right) - \left(1 - \frac{1}{\alpha}\right) = 1 - \left(1 - \frac{1}{\alpha}\right) = \frac{1}{\alpha}.$$

Since $|c| \geq 1$ by Step 1, we conclude that player i receives a total value of at least

$$\sum_{j \in R_i} v_{ij} = \sum_{j \in J_i : j \in J_{c,v}^i} v \geq \sum_{j \in J_i : j \in J_{c,v}^i} \frac{v}{|c|} \geq \frac{1}{\alpha}.$$

A visualization of this argument is given in Figure 2. ∎

2.2 Equivalence in the unrelated 2-value case In this section, we consider the two-value case of SANTACLAUS and MAKESPAN, where all resource values and job sizes are in $\{u, w, 0\}$ and $\{u, w, \infty\}$, respectively, with $u, w \geq 0$. Assume w.l.o.g. that $u \leq w$. We prove that there exists an approximation preserving equivalence between these two problems. We first restate the main result of this section, Theorem 1.2, for convenience.

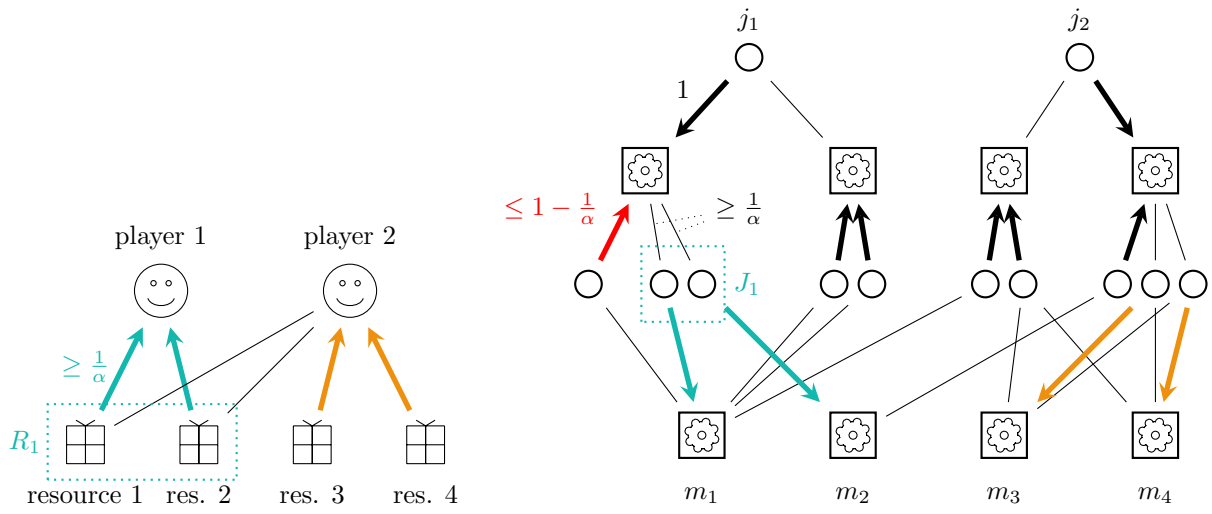
THEOREM 2.1. *For any $\alpha \geq 2$, there exists an α-approximation algorithm for two-value* SANTACLAUS *if and only if there exists a $(2 - 1/\alpha)$-approximation algorithm for two-value* MAKESPAN.

To prove this theorem, we prove two lemmas (Lemma 2.6 and Lemma 2.9) in the following subsections, one for each direction. Then, a standard binary search argument completes the proof.

Makespan to Santa Claus. For the direction from two-value MAKESPAN to two-value SANTACLAUS, we show that similar ideas as in the reduction of Lemma 2.1 also work in the direction from MAKESPAN to SANTACLAUS if we only have two job sizes. This requires some additional ideas to ensure that we do not need to introduce further resource values.

LEMMA 2.6. *Let I be an instance of the two-value* MAKESPAN *problem with* $\mathrm{OPT}(I) \leq 1$. *For any $\alpha \geq 2$, we can construct in polynomial time an instance I' of two-value* SANTACLAUS *such that, given an α-approximate solution for I', we can compute in polynomial time a solution for I with an objective value of at most $2 - 1/\alpha$.*

Fix an instance I of the two-value MAKESPAN problem with $\mathrm{OPT}(I) \leq 1$. First, we observe that we can w.l.o.g. assume that $w > \mathrm{OPT}(I)/2$. Otherwise, the algorithm by [25] gives us a solution of objective value at



(a) α-approximation in instance I.

(b) $(2 - 1/\alpha)$-approximation in instance I'.

Figure 2: Visualization of the argument for translating approximate solutions used in Lemma 2.5 applied to the example given in Figure 1.

most $\mathrm{OPT}(I) + w \leq 3/2 \cdot \mathrm{OPT}(I)$. Since $\alpha \geq 2$, this solution satisfies the lemma for every possible α. Thus, in the following we assume $w > \mathrm{OPT}(I)/2$.

We construct an instance I' of the two-value SantaClaus problem as follows:

1. Let $k = \min\{\lfloor 1/u \rfloor, n\}$, where n is the number of jobs in I. That is, k denotes the maximal number of small jobs that can be assigned to a single machine in an optimal solution with $\mathrm{OPT}(I) \leq 1$. By our assumption on the size w, we have that at most one big job can be placed on a single machine.

2. For every machine i we introduce a machine-player q_i, one (large) resource r_i, and k (small) resources $r_i^1, \ldots, r_i^k$. The value of the large resource r_i for player q_i is equal to w, the value of a small resource r_i^ℓ for player q_i is equal to u.

3. For every job j, we introduce a job-player $\hat{q}_j$. Furthermore, for every machine i, we set the value of resource r_i for $\hat{q}_j$ to w if $p_{ij} = w$ and to 0 otherwise. For a small resource r_i^ℓ we set the value for $\hat{q}_j$ to w if $p_{ij} = u$ and to 0 otherwise.

Note that in I', every machine-player q_i has only values in $\{0, u, w\}$, and every job-player $\hat{q}_j$ has only values in $\{0, w\}$. Thus, I' is a two-value SantaClaus instance. Further, for every machine the number of introduced resources is at most the total number of jobs in I plus one, asserting that I' is of polynomial size. An illustration of this construction is depicted in Figure 3.

We continue by proving two auxiliary lemmas on the constructed instance I' that together imply Lemma 2.6. To this end, let $t = w + k \cdot u - 1$ and note that $t \leq 1$ holds by construction. Also, observe that

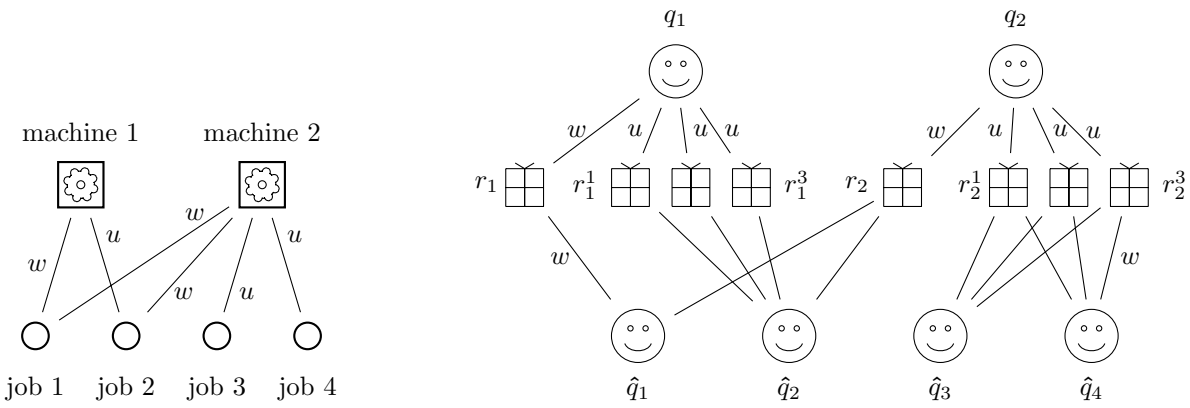
$$t = w + k \cdot u - 1 \leq w + k \cdot u - k \cdot u = w,$$

which implies $w \geq t$. Using this, we prove the following lemma.

Lemma 2.7. $\mathrm{OPT}(I') \geq t$.

Proof. Fix an optimal solution of I and recall that we assume $\mathrm{OPT}(I) \leq 1$. In the following we construct a solution for I'.

Fix a machine i of instance I. If the given solution assigns a job j of size w to i, we assign resource r_i to job-player $\hat{q}_j$. If the given solution assigns a job j of size u to machine i, we assign an arbitrary unassigned small resource r_i^ℓ to job-player $\hat{q}_j$. All yet unassigned resources are assigned to their corresponding machine-players.



(a) Two-value MAKESPAN instance I.

(b) Two-value SANTACLAUS instance I'.

Figure 3: The construction used in Lemma 2.6. In both pictures an edge indicates that an item has a non-trivial value for an entity. Here, k is equal to 3.

We continue by separately proving that each player in instance I' receives a value of at least t, which implies the lemma.

First, consider the job-players. Since every job j is assigned to exactly one machine in I, the job-player $\hat{q}_j$ receives exactly one resource in the constructed solution for I'. These resources have value w for those players, giving them a sufficiently large value of $w \geq t$.

Next, we consider the machine-players. Fix a machine i. By our assumption that $\mathrm{OPT}(I) \leq 1$, every machine i receives jobs of total size at most 1 in the solution for instance I. For our constructed solution to instance I', this means that the subset N_i of resources $r_i, r_i^1, \ldots, r_i^k$ that are *not* assigned to machine-player q_i satisfies $v_i(N_i) = \sum_{r \in N_i} v_{r,q_i} \leq 1$. This implies that the value assigned to machine-player i is a least

$$w + k \cdot u - v_i(N_i) \geq w + k \cdot u - 1 = t,$$

which implies $\mathrm{OPT}(I') \geq t$. ∎

We conclude the proof of Lemma 2.6 by showing the following lemma.

LEMMA 2.8. *For any $\alpha \geq 2$, given an α-approximate solution for I', we can construct in polynomial time a solution for I where every machine has a makespan of at most $2 - 1/\alpha$.*

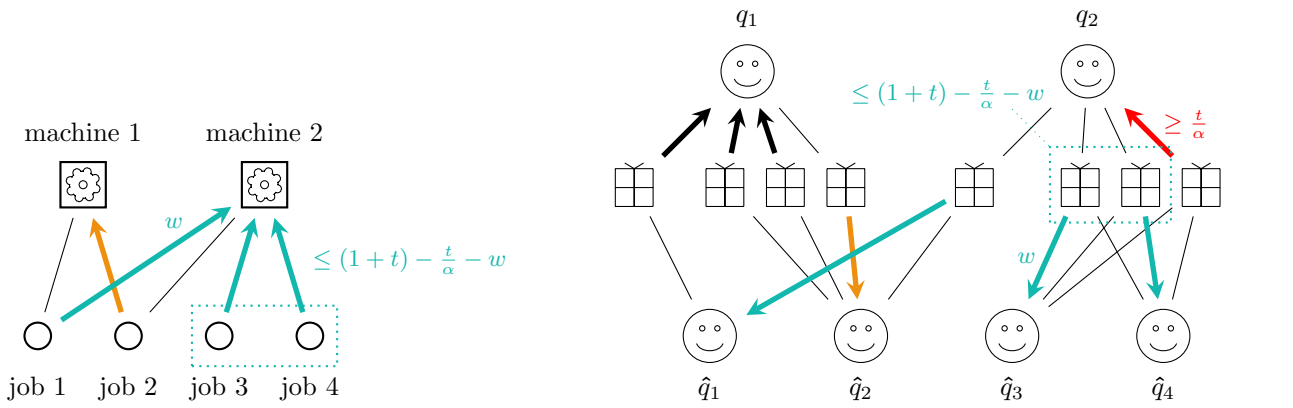
Proof. Consider an α-approximate solution for instance I'. Such a solution must assign resources of value at least $\mathrm{OPT}(I')/\alpha$ to each player. By the previous lemma, $\mathrm{OPT}(I')/\alpha \geq t/\alpha$.

Clearly, in the given solution for I' every job-player $\hat{q}_j$ receives at least one resource, as otherwise the objective value would be zero. We modify the given solution in a way such that each job-player receives exactly one resource. If a job-player receives more than one resource, we select an arbitrary resource and reassign all other resources to their corresponding machine-players. Since the single resource that remains assigned to a job-player gives it a value of $w \geq t$, the modified solution for I' still has an objective value of at least t/α.

Now, we can construct a solution for I as follows. If one of the resources $r_i, r_i^1, \ldots, r_i^k$ belonging to machine i has been assigned to a job-player $\hat{q}_j$, we assign job j to machine i. By the above assumption this assignment is well-defined.

It remains to argue about the load of every machine in I. Thus, fix a machine i. In the solution to instance I', the corresponding machine-player receives resources of value at least t/α. This means that the subset N_i of resources $r_i, r_i^1, \ldots, r_i^k$ that are *not* assigned to machine-player q_i satisfies

$$v_i(N_i) \leq w + k \cdot u - \frac{t}{\alpha} = 1 + t - \frac{t}{\alpha}.$$



(a) $(2-1/\alpha)$-approximation in instance I.

(b) α-approximation in instance I'.

Figure 4: Visualization of the argument for translating approximate solutions used in Lemma 2.8 applied to the example given in Figure 3.

Since by construction $t \leq 1$, we have $t - t/\alpha \leq 1 - 1/\alpha$, which implies $v_i(N_i) \leq 2 - 1/\alpha$. We conclude the proof by observing that, by construction, the makespan of machine i in the constructed solution for I is exactly equal to $v_i(N_i) \leq 2 - 1/\alpha$.

A visualization of the argument used in Lemma 2.8 is given in Figure 4. ∎

Santa Claus to makespan. For the direction from two-value SantaClaus to two-value Makespan, we show that we can apply a similar reduction as in Lemma 2.1. We do so by reducing the given SantaClaus instance to an instance where the reduction does not introduce a third value. Further, we observe that, in the two-value case, we only have a polynomial number of relevant configurations, so we do not have to reduce this number and, thus, do not lose the additional factor of $1 + \epsilon$.

Lemma 2.9. *Let I be an instance of the two-value* SantaClaus *problem with* $\mathrm{OPT}(I) \geq 1$. *For any $\alpha \geq 2$, we can construct in polynomial time an instance I' of two-value* Makespan *such that, given an $(2-1/\alpha)$-approximate solution for I', we can compute in polynomial time a solution for I with an objective value of at least $1/\alpha$.*

Proof. Let I be an instance of the two-value SantaClaus problem with $\mathrm{OPT}(I) \geq 1$ and $v_{ij} \in \{0, u, w\}$. We assume w.l.o.g. that $u \leq w$. We consider three exhaustive cases.

In the first case we assume that $w < \mathrm{OPT}(I)/\alpha$. Then, we can use the algorithm of Bezakova and Dani [8] to compute in polynomial time a solution in which every player receives a total value of at least $\mathrm{OPT}(I) - v_{\max} = \mathrm{OPT}(I) - w > (1 - 1/\alpha)\mathrm{OPT}(I) \geq \mathrm{OPT}(I)/\alpha$, using $\alpha \geq 2$.

In the second case we assume that $w \geq \mathrm{OPT}(I)/\alpha$ and that there is an optimal solution for I in which every player i receives a resource j of value $v_{ij} = w$. Then, we can essentially set u to 0 and compute a solution where every player receives a resource of value w by solving a bipartite matching problem. Since every player receives a value of at least $w \geq \mathrm{OPT}(I)/\alpha \geq 1/\alpha$, we are done.

In the last case we assume that $w \geq \mathrm{OPT}(I)/\alpha$ and that in every optimal solution for I there is some player i which does not receive a resource j of value $v_{ij} = w$. We can conclude that such a player must receive at least $b = \lceil 1/u \rceil$ many resources j' for which she has a value of $v_{ij'} = u$, because $\mathrm{OPT}(I) \geq 1$. Then, we construct a new instance I' by copying I and adjusting the resource values to $w' = 1$ and $u' = 1/b$. Observe that $\mathrm{OPT}(I') \geq 1$ and that any solution for I' in which every player either receives a resource of value 1 or b resources of value $1/b$ gives an objective value of at least 1. We can therefore define a collection of configurations $\mathcal{C}$ in which every player has these two configurations. Then, we can use Lemma 2.3 (by noting that in the constructed Makespan instance every job has either size 1 or $1/b$) to compute a solution for I' in which every player receives a total value of at least $1/\alpha$. Since $u \geq u'$ and $w \geq \mathrm{OPT}(I)/\alpha$, this means that we can use the same solution for instance I and guarantee that every player receives a total value of at least $1/\alpha$. ∎

3 Reduction from restricted matroid Santa Claus to the two-value case

The goal of this section is to prove the following lemma, which allows us to heavily reduce restricted resource-matroid SantaClaus instances to instances with only one matroid and one polymatroid.

Lemma 3.1. *For any $\alpha \geq 2$, if there is a polynomial-time algorithm that, given an instance of restricted two-value resource-matroid* SantaClaus *problem with one matroid of value $v_1 = \infty$ and one polymatroid of value $v_2 = 1$ and a number $b \in \mathbb{N}$, finds a solution of value at least b or determines that there is no solution of value αb, then there is also:*

1. *a polynomial-time α-approximation algorithm for (any instance of) the restricted two-value resource-matroid* SantaClaus *problem and*
2. *a polynomial-time 2α-approximation algorithm for the restricted resource-matroid* SantaClaus *problem.*

Before proving this lemma, we start this section with a rounding theorem which can be obtained by a variant of standard arguments in the scheduling literature (see [30]). In this section, we will often refer to a standard relaxation of the problem, called the *assignment LP*. In this LP, we have one variable $x_j(i)$ for each pair resource j and player i. In the resource-matroid SantaClaus problem, the relaxation can be written as follows.

$$\sum_{j \in R} x_j(i) \cdot v_{ij} \geq T \quad \forall i \in P$$
$$x_j \in \mathcal{B}(\mathcal{P}_j) \quad \forall j \in R$$
$$x \geq 0 \ .$$

Remark that we do not assume that we are in the restricted assignment setting, as our rounding theorems will apply even in the setting of unrelated resource values. In Appendix C we prove the equivalent statement of the following theorem for the unrelated job-matroid Makespan problem.

Theorem 3.1. *Given a fractional assignment x of resources to players which is feasible for the assignment LP (with parameter T) of some instance I of the unrelated resource-matroid* SantaClaus *problem, we can obtain, in polynomial time, a feasible integral assignment of resources to players of value at least $T - \max_{i \in P, j \in R} v_{ij}$.*

Proof. For each player i, we order the resources in non-increasing order of value. We will now define $\sigma_i(j)$ as the index corresponding to the resource that appears in j-th position in that order defined for player i. By definition we have that $v_{i\sigma_i(1)} \geq v_{i\sigma_i(2)} \geq \cdots \geq v_{i\sigma_i(n)}$ for any player i. W.l.o.g. we also assume that $v_{i\sigma_i(n)} = 0$ for any player i. We denote the polymatroids associated to the resources as $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$, corresponding to the submodular functions $f_1, f_2, \ldots, f_n$. Given the fractional assignment x, we will create a feasible fractional solution x' to a certain polymatroid intersection problem. We define the two polymatroids using a bipartite graph as follows. On the left-hand side, we have a set of vertices W with one vertex w_j for each resource j, and on the right-hand side we have a set of vertices U with one vertex u_{ij} for each player i and the resource j' which appears in jth position in the ordering of player i. The edge set will be denoted by E, and both polymatroids will have E as a ground set. We set $E = \{(w_{\sigma_i(j)}, u_{ij})\}_{i \in P, j \in R} \cup \{(w_{\sigma_i(j)}, u_{i(j+1)})\}_{i \in P, j \in R \setminus \{n\}}$. For some edge $e \in E$, we denote by e_w the resource corresponding to its left-hand side vertex, and by e_u the player corresponding to its right-hand side endpoint. The first polymatroid $\mathcal{P}'_1$ will be associated with the submodular function

$$f_1(S) := \sum_{j=1}^{n} f_j \left(\bigcup_{e \in S : e_w = j} e_u \right) \ .$$

The second polymatroid $\mathcal{P}'_2$ will be defined using the right-hand side vertices in our graph. Each vertex $u \in U$ will have some degree constraint $d(u)$ and the submodular function f_2 is simply defined as

$$f_2(S) := \sum_{e=(w,u) \in S} d(u) \ .$$

We define the degree constraints using the following process for each player i in instance I. Let us fix one player i. We start with

$$d(u_{i1}) = \left\lfloor x_{\sigma_i(1)}(i) \right\rfloor \ ,$$

and we define the *remainder* $R_1 := x_{\sigma_i(1)}(i) - d(u_{i1})$ (for ease of notation we define $R_0 = 0$). Then we define recursively the degree constraint $d(u_{ij})$ and remainders as follows.

$$d(u_{ij}) := \left\lfloor R_{j-1} + x_{\sigma_i(j)}(i) \right\rfloor \ , \text{ and}$$
$$R_j := \{R_{j-1} + x_{\sigma_i(j)}(i)\} = R_{j-1} + x_{\sigma_i(j)}(i) - \left\lfloor R_{j-1} + x_{\sigma_i(j)}(i) \right\rfloor = R_{j-1} + \{x_{\sigma_i(j)}(i)\} - \left\lfloor R_{j-1} + \{x_{\sigma_i(j)}(i)\} \right\rfloor \ .$$

By construction, it is clear that the fractional assignment x in instance I can be transformed into a feasible fractional solution to our polymatroid intersection problem. To see this, fix a player i. Then the first resource (in the order defined by player i) can be assigned to vertex u_{i1} up to an extent of $\left\lfloor x_{\sigma_i(1)}(i) \right\rfloor$, which is represented in our graph as taking $\left\lfloor x_{\sigma_i(1)}(i) \right\rfloor$ copies of the edge $(w_{\sigma_i(1)}, u_{i1})$. The remaining fraction of $x_{\sigma_i(1)}(i)$ can be assigned to player i by taking the edge $(w_{\sigma_i(1)}, u_{i2})$ fractionally by some amount $\{x_{\sigma_i(1)}(i)\}$. Then we move on to the second resource, and we take the edge (w_2, u_{i2}) by the maximal amount possible until the degree constraint on vertex u_{i2} becomes tight. By construction, we see that there might be some leftover of the value $x_{\sigma_i(2)}(i)$ which is precisely equal to the number R_2 in our construction. We continue this assignment until the last resource. Note that our definition of remainder R_j is precisely this small leftover of $x_{\sigma_i(j)}(i)$ that carries over to the edge going to vertex $u_{i(j+1)}$ (note that this remainder is always less than 1). There is one slight caveat at the end is that some small amount of the fractional assignment $x_{\sigma_i(n)}(i)$ might be thrown away, but as we will see, this does not hurt our purpose because we assume that $v_{\sigma_i(n)} = 0$.

Note that this fractional solution to our polymatroid intersection problem makes all the degree constraints on the right-hand side tight. By integrality of the polymatroid intersection polytope (see Chapters 46-47 in [27]) there exists an integral solution which also makes all the degree constraints on the right-hand side tight (and we can find it in polynomial time by finding the maximum cardinality multiset of edges which belongs to the polymatroids intersection).

It is easy to see that this integral solution corresponds to an integral assignment of resources to players in the instance I, in which each player i receives a value of at least

$$\sum_{j=1}^{n} \left\lfloor R_{j-1} + x_{\sigma_i(j)}(i) \right\rfloor v_{\sigma_i(j)} \ .$$

Let us compute the difference in objective Δ_i with the fractional solution. We have that

$$\Delta_i \leq \sum_{j=1}^{n} (x_{\sigma_i(j)}(i) - \left\lfloor R_{j-1} + x_{\sigma_i(j)}(i) \right\rfloor) v_{\sigma_i(j)} + v_{\sigma_i(n)} = \sum_{j=1}^{n} (\{x_{\sigma_i(j)}(i)\} - \left\lfloor R_{j-1} + \{x_{\sigma_i(j)}(i)\} \right\rfloor) v_{\sigma_i(j)} \ .$$

Looking at each term individually, we notice that either $\left\lfloor R_{j-1} + \{x_{\sigma_i(j)}(i)\} \right\rfloor = 0$, or that $\left\lfloor R_{j-1} + \{x_{\sigma_i(j)}(i)\} \right\rfloor = 1$. In the first case the next remainder R_j is equal to $R_{j-1} + \{x_{\sigma_i(j)}(i)\} < 1$. In the second case, we have that

$$R_j = R_{j-1} + \{x_{\sigma_i(j)}(i)\} - \left\lfloor R_{j-1} + \{x_{\sigma_i(j)}(i)\} \right\rfloor = R_{j-1} + \{x_{\sigma_i(j)}(i)\} - 1 \ .$$

Let us denote by R' the set of all indices where the second case happens. Then we can write

$$\Delta_i \leq \sum_{j=1}^{n} \{x_{\sigma_i(j)}(i)\} v_{\sigma_i(j)} - \sum_{j \in R'} v_{\sigma_i(j)} \ .$$

Now note that if $j \notin R'$, then $\{x_{\sigma_i(j)}(i)\} = R_j - R_{j-1}$, and that if $j \in R'$ then $\{x_{\sigma_i(j)}(i)\} = 1 + R_j - R_{j-1}$. Using this observation, we obtain

$$\Delta_i \leq \sum_{j=1}^{n} (R_j - R_{j-1}) v_{\sigma_i(j)} + \sum_{j \in R'} v_{\sigma_i(j)} - \sum_{j \in R'} v_{\sigma_i(j)} \leq \sum_{j=1}^{n} (R_j - R_{j-1}) v_{\sigma_i(j)} = \sum_{j=1}^{n-1} (v_{\sigma_i(j)} - v_{\sigma_i(j+1)}) R_j \leq v_{\sigma_i(1)} \ ,$$

where we use the fact that $R_0 = v_{\sigma_i(n)} = 0$, and $R_j \leq 1$ for all j. ∎

We finally prove Lemma 3.1.

Proof. We start by proving the first result of the lemma. Let u, w (with $w \geq u$) be the two sizes of the instance I, and let f_u, f_w be the associated submodular functions.

Now, we have three possible cases. If $\mathrm{OPT}(I)/\alpha \leq u$, then it suffices to give at least one resource to any player to obtain an α-approximate solution. This can be checked in polynomial time (see Chapter 42 in [27]). If $u < \mathrm{OPT}(I)/\alpha \leq w$, then, in an α-approximate solution, it suffices to give to each player either one resource of value w or $\mathrm{OPT}(I)/(\alpha u)$ resources of value u. We define a new instance I_2 with one matroid of value $v_1 = \infty$ and one polymatroid of value $v_2 = 1$. The independent sets of the matroid are the sets of players which can be covered by at least one resource of value w each in the original instance. The polymatroid is defined by the submodular function $f_2(S) := f_u(S)$. Clearly, in this case, we have that $\mathrm{OPT}(I_2) \geq \mathrm{OPT}(I)/u$, and a solution of value t in instance I_2 can immediately be translated into a solution of value $\min\{\mathrm{OPT}(I)/\alpha, tu\}$ in the original instance. So an α-approximation on instance I_2 gives us an α-approximation on the original instance. In the last case where $\mathrm{OPT}(I)/\alpha > w$, we first assume without loss of generality that u, w are integers (by appropriate scaling). Then we define a polymatroid $\mathcal{P}_3$ with the submodular function $f_3(S) := u \cdot f_u(S) + w \cdot f_w(S)$; this should be thought of splitting the resources of value w (respectively u) into w (respectively u) individual resources of value 1 each. The biggest b such that $b \cdot E \in \mathcal{P}_3$ (where E is the set of all players and $b \cdot E$ is the $|E|$ dimensional vector with all entries equal to b) can be found in polynomial time, since we simply need to minimize a submodular function (see Chapter 45 in [27]). This gives us a fractional solution to the original instance of objective value $\mathrm{OPT}(I)$. Using Theorem 3.1, we can round (in polynomial time) this fractional solution into an integral solution of objective value $\mathrm{OPT}(I) - \max\{u, w\} \geq \mathrm{OPT}(I) \cdot (1 - 1/\alpha) \geq \mathrm{OPT}(I)/\alpha$ (using $\alpha \geq 2$), which concludes the proof of the first point of the lemma.

For the second point, by a standard guessing strategy (as explained in the beginning of Section 2), we can assume that we know the optimum value $\mathrm{OPT}(I)$. We call a resource j *heavy* if $v_j \geq \mathrm{OPT}(I)/(2\alpha)$, and *light* otherwise (let H and L be the set of heavy and light resources respectively). We then define an instance I' with one matroid of value ∞, whose independent sets are the sets of players which can be covered by at least one heavy resource each (this is a matroid by the matroid union theorem see Chapter 42 in [27]). Again, assuming that all values v_j are integers, we define one polymatroid of value 1 associated to the submodular function $f'(S) := \sum_{j \in L} v_j f_j(S)$. In this new instance, it is clear that $\mathrm{OPT}(I') \geq \mathrm{OPT}(I)$. Hence an α-approximate solution to instance I' can be transformed into an α-approximate solution to instance I, in which the heavy resources are assigned integrally, and the light resources fractionally. Using Theorem 3.1, we can round this fractional assignment into and integral assignment of value at least $\mathrm{OPT}(I)/\alpha - \max_{j \in L} v_j \geq \mathrm{OPT}(I)/\alpha - \mathrm{OPT}(I)/(2\alpha) = \mathrm{OPT}(I)/(2\alpha)$. ∎

4 Reductions between matroid allocation problems

We consider the restricted assignment setting of the matroid generalizations of SantaClaus and Makespan. The main result here is Theorem 1.3, which we restate here for convenience.

Theorem 4.1. *For any $\alpha \geq 2$, there exists a polynomial-time α-approximation algorithm for the restricted two-value resource-matroid* SantaClaus *problem if and only if there exists a polynomial-time $(2-1/\alpha)$-approximation algorithm for the restricted two-value job-matroid* Makespan *problem.*

We prove the theorem using the following two lemmas, one for each direction, and the same standard binary search framework as in Section 2. The key ideas for constructing instances and for transforming solutions in these reductions rely on polymatroid duality. Note that, by Proposition 1.1, we can w.l.o.g. assume that the number of resources and jobs, respectively, is exactly two.

Lemma 4.1. *Let $\alpha \geq 2$ and I be an instance of the restricted job-matroid* Makespan *problem with two jobs and $\mathrm{OPT}(I) \leq 1$. Then we can compute an instance I' of the restricted resource-matroid* SantaClaus *problem with two resources, such that, given an α-approximate solution for I', we can compute a solution for I with an objective value of at most $2 - 1/\alpha$.*

Proof. Given an instance I with $\mathrm{OPT}(I) \leq 1$ of the restricted job-matroid Makespan problem with machines E and two jobs with sizes $p_1, p_2 \geq 0$ and polymatroids $\mathcal{P}_1, \mathcal{P}_2$, we construct an instance I' of the restricted resource-matroid SantaClaus problem as follows.

Let $k_1 = \lfloor 1/p_1 \rfloor$ and let $k_2 = \lfloor 1/p_2 \rfloor$. We first consider the polymatroids $\mathcal{P}'_1 = \{x \in \mathcal{P}_1 : x(e) \leq k_1 \ \forall e \in E\}$ and $\mathcal{P}'_2 = \{x \in \mathcal{P}_2 : x(e) \leq k_2 \ \forall e \in E\}$. Let f'_1 and f'_2 be the associated submodular functions of these polymatroids. Since $\mathrm{OPT}(I) \leq 1$, any optimal solution $x_j \in \mathcal{B}(P_j)$ satisfies $x_j(e) \leq k_j$ for all $e \in E$, and therefore, $x_j \in \mathcal{B}(P'_j)$, for $j \in \{1,2\}$. Thus, $f_j(E) = x_j(E) = f'_j(E)$. Let $\overline{\mathcal{P}}_j$ be the dual polymatroid of $\mathcal{P}'_j$ with respect to the vector $k_j \cdot E$ (the vector of $\mathbb{Z}^E$ where all entries are equal to k_j), for $j \in \{1,2\}$. We compose instance I' using players E and two resources with polymatroids $\overline{\mathcal{P}}_1, \overline{\mathcal{P}}_2$ and resource values p_1, p_2.

Let $t = k_1 \cdot p_1 + k_2 \cdot p_2 - 1$. We show that $\mathrm{OPT}(I') \geq t$. Fix an optimal solution for I which selects bases $x_1 \in \mathcal{B}(\mathcal{P}_1)$ and $x_2 \in \mathcal{B}(\mathcal{P}_2)$. For each $j \in \{1,2\}$, we define a vector $\overline{x}_j$ with $\overline{x}_j(e) = k_j - x_j(e)$ for all $e \in E$, and conclude that $\overline{x}_j \in \mathcal{B}(\overline{\mathcal{P}}_j)$, because $x_j \in \mathcal{B}(\mathcal{P}'_j)$. This means that $\overline{x}_1$ and $\overline{x}_2$ are a feasible solution for I'. Using $\mathrm{OPT}(I) \leq 1$, for every player $e \in E$ it holds that

$$p_1 \cdot \overline{x}_1(e) + p_2 \cdot \overline{x}_2(e) = (1+t) - (p_1 \cdot x_1(e) + p_2 \cdot x_2(e)) = (1+t) - \mathrm{OPT}(I) \geq t,$$

showing that $\mathrm{OPT}(I') \geq t$.

We finally prove the stated bound on the objective value of an approximate solution. Fix an α-approximate solution for I' which selects bases $\overline{y}_1 \in \overline{\mathcal{P}}_1$ and $\overline{y}_2 \in \overline{\mathcal{P}}_2$. We construct an approximate solution $y_j \in \mathcal{B}(\mathcal{P}_j)$ for instance I by setting $y_j(e) = k_j - \overline{y}_j(e)$ for every $e \in E$ and $j \in \{1,2\}$. The construction of the dual polymatroid $\overline{\mathcal{P}}_j$ implies $y_j \in \mathcal{B}(\mathcal{P}'_j)$ for $j \in \{1,2\}$. We further have $y_j \in \mathcal{B}(\mathcal{P}_j)$, because $\mathcal{P}'_j \subseteq \mathcal{P}_j$ and $y_j(E) = f'_j(E) = f_j(E)$. Moreover, for every machine $e \in E$ it holds that

$$p_1 \cdot y_1(e) + p_2 \cdot y_2(e) = (1+t) - (p_1 \cdot \overline{y}_1(e) + p_2 \cdot \overline{y}_2(e)) \leq (1+t) - \frac{1}{\alpha} \cdot \mathrm{OPT}(I') \leq 1 + t - \frac{t}{\alpha}.$$

Since by construction $t \leq 1$, we have $t - t/\alpha \leq 1 - 1/\alpha$, which implies that the makespan of the constructed solution (y_1, y_2) is at most $2 - 1/\alpha$. ∎

The second direction can be shown with the same proof idea and some additional tweaks specific to the direction.

LEMMA 4.2. *Let $\alpha \geq 2$ and I be an instance of the restricted resource-matroid* SANTACLAUS *problem with two resources and $\mathrm{OPT}(I) \geq 1$. Then we can compute an instance I' of the restricted job-matroid* MAKESPAN *problem with two jobs, such that, given a $(2 - 1/\alpha)$-approximate solution for I', we can compute a solution for I with an objective value of at least $1/\alpha$.*

Proof. Let I be an instance of the restricted resource-matroid SANTACLAUS problem with two resources with associated polymatroids $\mathcal{P}_1, \mathcal{P}_2$ and values $v_1, v_2 \geq 0$ such that $\mathrm{OPT}(I) \geq 1$. By Lemma 3.1 we can assume that we are given a simplified case. For convenience, we slightly reformulate it as follows: given $v_1 = 1$ (instead of ∞), $v_2 = 1/b \leq v_1$ (instead of 1) for $b \in \mathbb{N}$, and $\mathrm{OPT}(I) \geq 1$, find a solution of value at least $1/\alpha$.

Consider the polymatroids $\mathcal{P}'_1 = \{x \in \mathcal{P}_1 : x(e) \leq 1 \ \forall e \in E\}$ and $\mathcal{P}'_2 = \{x \in \mathcal{P}_2 : x(e) \leq b \ \forall e \in E\}$. Let $\overline{\mathcal{P}}_1$ be the dual polymatroid of $\mathcal{P}'_1$ with respect to the vector $1 \cdot E$, and let $\overline{\mathcal{P}}_2$ be the dual polymatroid of $\mathcal{P}'_2$ with respect to the vector $b \cdot E$. We compose an instance I' of job-matroid MAKESPAN using machines E, one job of size $p_1 = 1$ with polymatroid $\overline{\mathcal{P}}_1$ and one job of size $p_2 = 1/b$ with polymatroid $\overline{\mathcal{P}}_2$.

We now show that $\mathrm{OPT}(I') \leq 1$. Fix an optimal solution for I which selects bases $x_1 \in \mathcal{B}(\mathcal{P}_1)$ and $x_2 \in \mathcal{B}(\mathcal{P}_2)$. Since $\mathrm{OPT}(I) \geq 1$, we have $v_1 \cdot x_1(e) + v_2 \cdot x_2(e) \geq 1$ for every $e \in E$. Since $v_1 = 1$ and $v_2 = 1/b$, this implies $x_1(e) + (1/b) \cdot x_2(e) \geq 1$. Our goal is to dualize bases of $\mathcal{P}'_1$ and $\mathcal{P}'_2$ to obtain bases of $\overline{\mathcal{P}}_1$ and $\overline{\mathcal{P}}_2$, which are feasible for I'. To this end, we first construct vectors $x'_1 \in \mathcal{P}'_1$ and $x'_2 \in \mathcal{P}'_2$ such that $x'_1(e) + (1/b) \cdot x'_2(e) \geq 1$ for all $e \in E$. This can be done by restricting x_1 to values of at most 1 and x_2 to values of at most b, and then selecting any bases which dominate these intermediate vectors. Now, we define $\overline{x}_1(e) = 1 - x'_1(e)$ and $\overline{x}_2(e) = b - x'_2(e)$ for all $e \in E$. By the construction of $\overline{\mathcal{P}}_1$ and $\overline{\mathcal{P}}_2$, this solution is feasible for I', i.e., $\overline{x}_j \in \mathcal{B}(\overline{\mathcal{P}}_j)$ for $j \in \{1,2\}$. We further have for every machine $e \in E$

$$\overline{x}_1(e) + \frac{1}{b} \cdot \overline{x}_2(e) = 2 - \left(x'_1(e) + \frac{1}{b} \cdot x'_2(e) \right) \leq 1,$$

showing that $\mathrm{OPT}(I') \leq 1$.

We finally prove the stated bound on the objective value of an approximate solution. Fix a $(2-1/\alpha)$-approximate solution for I' which selects bases $\overline{y}_1 \in \mathcal{B}(\overline{\mathcal{P}}_1)$ and $\overline{y}_2 \in \mathcal{B}(\overline{\mathcal{P}}_2)$. We construct an approximate solution for I by defining $y_1'(e) = 1 - \overline{y}_1(e)$ and $y_2'(e) = b - \overline{y}_2(e)$ for every $e \in E$, meaning that $y_j' \in \mathcal{B}(\mathcal{P}_j')$, and then choose an arbitrary basis $y_j \in \mathcal{B}(\mathcal{P}_j)$ which dominates y_j', for $j \in \{1,2\}$. We have for every player $e \in E$

$$y_1(e) + \frac{1}{b} \cdot y_2(e) \geq y_1'(e) + \frac{1}{b} \cdot y_2'(e) = 2 - \left(\overline{y}_1(e) + \frac{1}{b} \cdot \overline{y}_2(e)\right) \geq 2 - \left(2 - \frac{1}{\alpha}\right) \cdot \mathrm{OPT}(I') \geq \frac{1}{\alpha}.$$

Since $v_2 = 1/b$ and $v_1 = 1$, we conclude $v_1 \cdot y_1(e) + v_2 \cdot y_2(e) \geq 1/\alpha$ for every $e \in E$, which implies that the value of the constructed solution y_1 and y_2 is at least $1/\alpha$. ∎

5 Local search algorithm

In this section we present our algorithm for Theorem 1.4 that finds an $(8+\epsilon)$-approximation for the restricted resource-matroid SantaClaus problem and a $(4+\epsilon)$-approximation in the case of two values.

We will show in Lemma 3.1 that it suffices to solve the following problem with $\alpha = 4+\epsilon$. Given a matroid $\mathcal{M} = (E, \mathcal{I})$ and a polymatroid $\mathcal{P}$ over the same set of elements as well as some $b \in \mathbb{N}$, find some $I_M \in \mathcal{I}$ and $y \in \mathcal{P}$ such that for every $i \in E$ we have $i \in I_M$ or $y(i) \geq b$, or determine that no solution exists for αb. Before we move to the algorithm, we define some specific notation used throughout the section.

Notation. Recall that for some $X \subseteq E$, we write $b \cdot X$ as the vector $y \in \mathbb{Z}^E$ with $y(i) = b$ for $i \in X$ and $y(i) = 0$ for $i \notin X$. Contracting a set $X \subseteq E$ of a matroid $\mathcal{M}$ defines a new matroid $\mathcal{M}/X$ obtained from restricting the elements to $E \setminus X$ and defining the rank function as $r(Y \mid X) = r(Y \cup X) - r(X)$. Naturally, the independent sets of the contracted matroid form all the sets that together with *any* independent set in S are independent in the original matroid. We use contraction primarily to fix some elements in the matroid. We need a similar notion for the polymatroid $\mathcal{P}$ defined by the submodular function f. However, while a single element $i \in E$ has rank function $r(i) \leq 1$ in a matroid, the value $f(i)$ could be arbitrarily large. In particular, contracting a set using $f(Y \mid X)$ we may reserve more resources for X than intuitively necessary (recall we only need to cover elements with the polymatroid b times). Hence, we need to use a more sophisticated approach here. First, consider the polymatroid $\mathcal{P}' = \{y \in \mathcal{P} : y(i) \leq b \ \forall i \in X\}$, i.e., a restriction on the multiplicity of each element in X within the polymatroid. Let, f' be the submodular function defining $\mathcal{P}'$. After this transformation we can use $f'(Y \mid X)$ without the aforementioned issues. We introduce the short notation

$$f(Y \mid b \cdot X) = f'(Y \mid X)\ .$$

Note that $f(Y \mid b \cdot X)$ behaves as one would expect in the sense of decreasing marginal returns, see Lemma 5.1. We will define both $r(Y \mid X)$ and $f(Y \mid b \cdot X)$ on all $X \subseteq E$ instead of only $Y \subseteq E \setminus X$. More precisely, $r(Y \mid X) = r(Y \cup X) - r(X) = r((Y \setminus X) \cup X) - r(X) = r(Y \setminus X \mid X)$ gives a natural extension although clearly the elements in X behave trivially. Similar to this, we extend $f(Y \mid b \cdot X)$ to $Y \cap X \neq \emptyset$ by defining $f(Y \mid b \cdot X) = f(Y \setminus X \mid b \cdot X)$.

Framework. We move to a further variation of the problem, which resembles an augmentation framework similar to matroid partition problems, where we are given a partial solution that we then extend. The algorithm itself is defined using recursion with the following interface.

Input. Matroid $\mathcal{M} = (E, \mathcal{I})$ with rank function r, polymatroid $\mathcal{P} \subseteq \mathbb{Z}^E$ with function f, both over the same elements, and a number $b \in \mathbb{N}$.

Further, disjoint sets $I_M \in \mathcal{I}, b \cdot I_P \in \mathcal{P}, B_0 \subseteq E$. Finally, a partial order $\prec$ on B_0.

Output. Either an augmented solution $I_M' \in \mathcal{I}$ and $b \cdot I_P' \in \mathcal{P}$ such that $I_M' \dot{\cup} I_P' \supseteq I_M \cup I_P$ and $|I_M' \cap B_0| \geq \epsilon^2 |B_0|$ or "failure".

In case failure is returned, we provide a certificate that proves that no I_M^*, I_P^* can exist with $I_M^* \cup I_P^* \supseteq I_M \cup I_P$, $I_M^* \in \mathcal{I}$ and the stronger conditions $\alpha b \cdot I_P^* \in \mathcal{P}$ for $\alpha = 4 + \mathcal{O}(\epsilon)$ and $|I_M^* \cap B_0| \geq 3\epsilon |B_0|$. Details on the certificate follow in the analysis.

The partial order $\prec$ affects which elements of B_0 the algorithm tries first to add to I_M. The precise guarantees on the output are subtle, but important inside the recursion, see proof of Lemma 5.5.

We can apply this variant to solve our previous polymatroid problem as follows: we initialize I_P as the set of all elements $i \in E$ that have $r(i) = 0$. If $b \cdot I_P \notin \mathcal{P}$ it is clear that the optimum is smaller than b. Assuming $b \cdot I_P \in \mathcal{P}$ we set $\mathcal{I} = \emptyset$ and now extend $I_M \cup I_P$ one element at a time by calling the procedure above with $B_0 = \{i\}$ for some element $i \notin I_M \cup I_P$. Each time I_M and I_P will be changed, but end up covering an additional element. Repeating this at most $|E|$ times we either have a solution that covers all elements or some element i cannot be added, which certifies that no solution exists with i covered by the matroid. In this case we alter the rank function to $r(X) \leftarrow r(X - i)$, setting in particular $r(i) \leftarrow 0$. We then restart the whole procedure.

We will now describe how to solve this variant of the problem. First, we assume without loss of generality that $I_M \cup I_P \cup B_0 = E$ by simply dropping irrelevant elements from the input. As stated above, we want to add many elements of B_0 to I_M. In the trivial case that $r(B_0 \mid I_M) \geq \epsilon^2 |B_0|$, we add greedily as many elements of B_0 as possible to I_M (while maintaining $I_M \in \mathcal{I}$), which will result in $|I_M \cap B_0| \geq \epsilon^2 |B_0|$, and we terminate successfully. Otherwise, we will have to remove elements from I_M before we can add sufficiently many elements of B_0 to I_M. To this end, we will carefully construct a set of *addable elements* $A \subseteq I_M$, where the notion has historical reasons and comes from the idea that we want to "add" A to I_P. The procedure for creating A is deferred to later and here we summarize only its important properties. The existence of A will be guaranteed by the algorithm. Along with A we also create the set C with $A \subseteq C \subseteq I_M$, which contains more elements of I_M that are relevant for adding B_0 to I_M (but not all of them could be added to A). The relevant properties of A and C are as follows.

1. It holds that $2b \cdot A \in \mathcal{P}$, which means that in principle A could be added to I_P (and removed from I_M). Note that this does not take into account potential conflicts with other elements currently in I_P. Also remark that we are intentionally overprovisioning here, by using $2b$ instead of b.
2. Set A is maximal within C regarding the previous property. More specifically, $f(i \mid 2b \cdot A) < 2b$ for all $i \in C \setminus A$.
3. If we remove many elements of A from I_M, we are able to add many elements of B_0 to I_M. Specifically, we require that for every $R \subseteq A$ with $|R| \geq \epsilon |A|$ we have
$$r(B_0 \mid I_M \setminus R) \geq \epsilon^2 |B_0| \ .$$
We note that while this property initially holds, only a weaker version is maintained as the algorithm progresses. For more details see Lemma 5.5.
4. Set C should contain almost all elements that block elements of B_0 from being added to I_M. Formally, $r(B_0 \mid C) \leq 2\epsilon |B_0|$. In particular, a solution that covers many elements of B_0 with the matroid needs to cover a substantial amount of elements in C with the polymatroid.

Our new goal is to move many elements of A to I_P, which may not be possible immediately because of conflicting elements currently in I_P. We first characterize these elements: Define the *blocking elements* B as the set of all $i \in I_P$ such that $f(i \mid b \cdot (I_P \cup A - i)) < b$. It is intuitively clear that elements not in B are not relevant to adding A: assume we remove some elements of I_P and add some elements of A to it. Then afterwards all elements not in B can easily be added back to I_P (if they were removed), since each of their marginal values will still be at least b.

When an element in A can be added to I_P, we will not add it right away. Instead we will only place it in a set of *immediately addable elements* A_I. Only when we have enough of these elements to successfully terminate, we will add them to I_P. This is mainly for simplicity, i.e., to keep structures as static as possible during execution. We now repeatedly perform the first possible operation from the following:

1. If $f(i \mid b \cdot (I_P \cup A_I)) \geq b$ (equivalently, $b \cdot (I_P \cup A_I + i) \in \mathcal{P}$) for some $i \in A \setminus A_I$, add i to A_I.
2. If $|A_I| \geq \epsilon |A|$, add A_I to I_P, remove it from I_M, and greedily add as many elements of B_0 as possible to I_M, in the order given by $\prec$. We can then terminate successfully (Lemma 5.5).
3. If $|B| < \epsilon |B_0|$, return "failure".
4. If none of the above applies, we will recurse on B, which means that we try to move many elements of B to I_M so that they can be removed from I_P, hopefully allowing us to move elements of A to A_I. The details of the recursion follow towards the end of the section.

Construction of addable elements. We construct a series of disjoint sets $A_1, A_2, \ldots$ as follows: assume that $A_1, A_2, \ldots, A_{\ell-1}$ have already been created. We initialize $A_\ell = \emptyset$. Then repeat the following until exhaustion: if there exists an $i \in I_M \setminus (A_1 \cup A_2 \cup \cdots \cup A_\ell)$ with $r(i \mid B_0 \cup I_M \setminus A_\ell - i) = 0$ and

$$f(i \mid 2b \cdot (A_1 \cup A_2 \cup \cdots \cup A_\ell)) \geq 2b \ ,$$

then we add i to A_ℓ. If $|A_\ell| < \epsilon|B_0|$ we terminate with $A = A_1 \cup A_2 \cup \cdots \cup A_{\ell-1}$. Otherwise, we continue with the next set. When the construction of A is finalized, we define

$$C = B_0 \cup A \cup \{i \in I_M \setminus A : f(i \mid 2b \cdot A) < 2b\} \ .$$

Recursion. We will denote the input of the recursion using prime next to the symbol, e.g. E', B'_0, etc. The goal of the recursion is to move many elements of B to I_M. On the other hand, we want to keep our structures within B_0, A, and C largely intact. To this end, we first contract C from the matroid, which means that the recursion cannot remove elements of C from I_M. It may not be intuitively clear why this would be bad for us, but removing many elements of C does not necessarily allows us to add many elements of B_0 (comparable to Property 3 of the set A) for arbitrary elements of C. In any case, we want to avoid the complications related to such changes in C. Hence, we set

$$\begin{aligned} E' &= E \setminus C \ , \\ I'_M &= I_M \setminus C \ , \text{ and} \\ r'(X) &= r(X \mid C) \quad \forall X \subseteq E' \end{aligned}$$

which defines a new matroid $\mathcal{M}' = (E', \mathcal{I}')$. Regarding the polymatroid, we remove B from I_P so as to produce an input where sets B'_0, I'_M, and I'_P are disjoint. However, by contracting $b \cdot B$ from the polymatroid we make sure that we can add it back to the modified solution after the recursion has returned (at least for those elements that were not moved into I_M). Furthermore, we want to avoid that the recursion moves elements to I_P that hinder A from being added to the polymatroid. Hence, we contract $b \cdot A$ as well. This is achieved by setting

$$\begin{aligned} I'_P &= I_P \setminus B \text{ and} \\ f'(X) &= f(X \mid b \cdot (A \cup B)) \quad \forall X \subseteq E' \ . \end{aligned}$$

From f' we obtain the new polymatroid $\mathcal{P}'$. Finally, we define

$$B'_0 = B_0 \cup B$$

and extend $\prec$ by giving all elements of B lower priority than B_0. Intuitively, it does not hurt us to include B_0 in B'_0. If the recursion manages to move elements to B_0, this is only good for us. We prove in Lemma 5.7 that this indeed forms a valid input of the problem. If the recursive call returns failure, we return failure as well. Otherwise, we update as follows. Let I''_M and I''_P be the output of the recursion. First, we add back the previously removed C:

$$I_M \leftarrow C \cup I''_M \ .$$

As for I_P, we want to add back B except for those elements that were covered with the matroid in the recursive call. Thus,

$$I_P \leftarrow (B \setminus I''_M) \cup I''_P \ .$$

In Lemma 5.6 we show that the new I_M, I_P constitute again a feasible solution. If the returned set I''_M satisfies $|B_0 \cap I''_M| \geq \epsilon^2|B_0|$ we terminate successfully. Otherwise, it must hold that $|B \cap I''_M| \geq \epsilon^2|B|$, which intuitively means we made big progress in freeing B and is used in the running time analysis. Since I_P has changed, B may no longer correspond to its initial definition. Hence, we update B to again reflect the set of all $i \in I_P$ with $f(i \mid b \cdot (I_P \cup A - i)) < b$ according to the new set I_P.

5.1 Analysis

General properties of matroids and submodular functions.

LEMMA 5.1. *Let $g : E \to \mathbb{Z}_{\geq 0}$ be monotone submodular with $g(\emptyset) = 0$, $X' \subseteq X \subseteq E$, and $h' < h$. Then*

$$g(Y \mid h \cdot X') \geq g(Y \mid h \cdot X) \text{ for all } Y \subseteq E \setminus X \text{ and}$$
$$g(Y \mid h' \cdot X) \geq g(Y \mid h \cdot X) \text{ for all } Y \subseteq E \setminus X \ .$$

Proof. Recall that $g(Y \mid h \cdot X)$ is derived by first constructing a new monotone submodular function g' corresponding to the polymatroid defined by g but with entries of X bounded by h. For the first inequality, let g'' be the corresponding function with X' instead of X. Then

$$g(Y \mid h \cdot X') = g''(Y \mid X') = g''(Y \cup X') - g''(X') \\ \geq g'(Y \cup X') - g'(X') = g'(Y \mid X') \geq g'(Y \mid X) = g(Y \mid h \cdot X) \ .$$

Here we use that $g''(X') = g'(X')$. For the second inequality, let g''' be the submodular function for h' instead of h. Let $\mathcal{Q}$ be the polymatroid corresponding to g. Then there exists some $y''' \in \mathcal{Q}$ with $\text{supp}(y''') \subseteq X$, $y'''(i) \leq h'$ for all $i \in E$, and $y'''(X) = g'''(X)$. We define y' accordingly, except for g' instead of g'''. Here we may assume, by augmentation property of the polymatroid that $y'(i) \geq y'''(i)$ for all $i \in E$. The value $g(Y \mid h' \cdot X) = g'''(Y \mid X)$ is simply the largest element (by sum) in the polymatroid defined by restricting $\mathcal{Q}$ to $E \setminus X$ and replacing the submodular function by $g'''(Y') = g(Y' \cup X) - y'''_i(X)$. It is clear that this is at least as big as $g(Y \mid h \cdot X) = g'(Y \mid X)$, since the corresponding polymatroid here has a submodular function $g'(Y') = g(Y' \cup X) - y'_i(X)$ that is smaller or equal to g''' everywhere. ∎

LEMMA 5.2. *Let $g : E \to \mathbb{Z}_{\geq 0}$ be monotone submodular with $g(\emptyset) = 0$ and $X \subseteq E$. Define $Y = \{i \in X : g(i \mid h \cdot (X - i)) < h\}$. Then for every $i \in X$ we have $g(i \mid h \cdot (Y - i)) < h$ if and only if $i \in Y$.*

Proof. For any $i \in X$ with $g(i \mid h \cdot (Y - i)) < h$ we have $g(i \mid h \cdot (X - i)) < h$ by Lemma 5.1. This proves one direction. For the other direction, let $i \in X$ with $g(i \mid h \cdot (Y - i)) \geq h$. Further, let $\mathcal{Q}$ be the polymatroid corresponding to g and let $y \in \mathcal{Q}$ with $\text{supp}(y) \subseteq Y - i$ and $y(j) \leq h$ for all $j \in E$. Further, choose y such that $y(Y - i)$ is maximized. Since $g(i \mid h \cdot (Y - i)) \geq h$, we can extend y to y' which is equal for all $j \neq i$ and has $y'(i) = h$. Now for any $i' \in X \setminus Y$, since $g(i' \mid h \cdot Y) \geq g(i' \mid h \cdot (X - i')) \geq h$ we can repeat the same trick and increase the value of $y'(i')$ to h as well. It is easy to see that this can be continued to obtain $y'' \in \mathcal{Q}$ with $y''(j) = h$ for all $j \in X \setminus (Y - i)$ and $y''(j) = y(j)$ for all $j \in Y - i$.

It is clear that y'' when restricted to $X - i$ maximizes $y''(X - i)$ over all elements of $\mathcal{Q}$ with support $X - i$ and upper bound h: It maximizes the sum already on $Y - i$ and all other components are obviously the largest possible. Together with the fact that y'' with $y''(i) = h$ is in $\mathcal{Q}$, this implies that $g(i \mid h \cdot (X - i)) \geq h$. ∎

LEMMA 5.3. *Let $g : E \to \mathbb{Z}_{\geq 0}$ be monotone submodular with $g(\emptyset) = 0$ and $X' \subseteq X \subseteq E$. Further, let $g(i \mid h \cdot (X - i)) < h$ for all $i \in X'$. Then $g(X') \leq h|X|$ and strict inequality holds whenever $X' \neq \emptyset$.*

Proof. We use an induction over $|X'|$. For $X' = \emptyset$ the claim obviously holds. Now consider $X' \neq \emptyset$ and let $i \in X'$. There must be some $Y \subseteq X$ with $i \in Y$ such that $g(Y) < h|Y|$: assume otherwise and let $\mathcal{Q}$ be the polymatroid corresponding to g. Let $z \in \mathcal{Q}$ with $z(j) \leq h$ for all $j \in E$ and $z(i) = 0$, maximizing $z(E)$. It can easily be checked that z' with $z'(j) = z(j)$ for $j \neq i$ and $z'(i) = h$ is also in $\mathcal{Q}$. This however implies that $g(i \mid h \cdot (X - i)) \geq h$.

Having established that $g(Y) < h|Y|$ for some $Y \ni i$, we use the induction hypothesis on $X \setminus Y$, $X' \setminus Y$ and $g'(Y') := g(Y' \mid Y)$. It follows that $g(X' \setminus Y \mid Y) \leq h|X \setminus Y|$ and therefore

$$g(X') \leq g(Y) + g(X' \setminus Y \mid Y) < h|Y| + h|X \setminus Y| = h|X| \ .$$

This concludes the proof. ∎

Basic properties and invariants of the data structures. Note that after A, C, B, and A_I are initially created, we never change $A, C, I_M \cap C$ or $I_P \cap C$. The only dynamic sets are B, A_I, $I_M \setminus C$, and $I_P \setminus C$. Hence, for properties that rely solely on the fixed elements, it suffices to verify them at the time they were created.

Furthermore, the only place that makes potentially dangerous changes to B, $I_M \setminus C$, and $I_P \setminus C$ is the recursion. In the remainder, "at all times" means that a property should hold between any two of the four main operations that are performed repeatedly.

We will now verify the properties of the set of addable elements. Notice that Properties (1) and (2), that is, $2b \cdot A \in \mathcal{P}$ and $f(i \mid 2b \cdot A) < 2b$ for all $i \in C \setminus A$, hold trivially by construction.

LEMMA 5.4. *For the set of addable elements A and set C we have that $r(B_0 \mid C) \leq 2\epsilon|B_0|$.*

Proof. Let $A_1, A_2, \ldots, A_\ell$ be the sets created by the procedure. Consider the time that C is created. Here, it holds that $r(B_0 \mid I_M) < \epsilon^2|B_0| \leq \epsilon|B_0|$. Now assume towards contradiction that $r(B_0 \mid C) > 2\epsilon|B_0|$. Thus, we can find some $X \subseteq B_0$ with $|X| = r(B_0 \mid C)$ and $C \cup X \in \mathcal{I}$. After finalizing A, for all $i \in I_M \setminus (C \cup A_\ell)$ we have $r(i \mid B_0 \cup I_M \setminus A_\ell - i) = 1$. Thus, $X \cup I_M \setminus A_\ell \in \mathcal{I}$, which can be seen by adding to $C \cup X$ the elements from $I_M \setminus (A_\ell \cup C)$ one at a time (each having marginal value 1). Applying the matroid augmentation property on I_M, we can find a set $Y \subseteq B_0$ such that $Y \cup I_M \in \mathcal{I}$ and

$$|Y| = |X \cup I_M \setminus A_\ell| - |I_M| \geq |X| - |A_\ell| > \epsilon|B_0| \ ,$$

a contradiction. ∎

LEMMA 5.5. *For the set of addable elements A and set C we have that $r(B_0 \mid I_M \setminus R) \geq \epsilon^2|B_0| - |B_0 \cap I_M|$ for every $R \subseteq A$ with $|R| \geq \epsilon|A|$.*

Proof. We will first prove the statement for the time when A was created, but then we need to show that it also holds later. Let $A_1, A_2, \ldots, A_\ell$ be the sets created by the procedure and recall that $A = A_1 \cup \cdots \cup A_{\ell-1}$ and $A_j \geq \epsilon B_0$ for all $j < \ell$. Thus, there must be some A_j, $j < \ell$, with $|R \cap A_j| \geq \epsilon|A_j| \geq \epsilon^2|B_0|$. For I_M at the time of construction it holds that

$$r(B_0 \cup I_M \setminus (R \cap A_j)) \geq r(B_0 \cup I_M \setminus A_j) = r(B_0 \cup I_M) \geq |I_M| \ .$$

This is because A_j is constructed greedily from elements that do not decrease the rank. Hence,

$$\begin{aligned} r(B_0 \mid I_M \setminus R) &\geq r(B_0 \mid I_M \setminus (R \cap A_j)) \\ &= r(B_0 \cup I_M \setminus (R \cap A_j)) - r(I_M \setminus (R \cap A_j)) \geq |I_M| - |I_M| + |R \cap A_j| \geq \epsilon^2|B_0| \ . \end{aligned}$$

We will now study the effects of I_M changing throughout the algorithm. Essentially, when an element of B_0 is added to I_M then $r(B_0 \mid I_M \setminus R)$ may decrease by 1, otherwise it does not change. Note that we can view the changes made by the algorithm (or its recursive calls) to I_M as a sequence of single insertions or deletions. More precisely, there exists a sequence of sets Let $I_1, I_2, \ldots, I_k \in \mathcal{I}$, where I_k is the current state of I_M that we want to analyze, I_1 is the initial state, and I_{h+1} is derived from I_h either by deletion or addition of a single element. Further, whenever $I_{h+1} = I_h + i$ for some $i \notin I_h$, then we know that $I_h + j \notin \mathcal{I}$ for all $j \notin I_h$ with $j \prec i$. Finally, once an element of B_0 is added to some I_h, it remains in I_M, i.e., it is also in $I_{h+1}, \ldots, I_k$. All of these properties are observations that follow easily from the definition of the algorithm.

Let $1 \leq h < k$, $R \subseteq C \subseteq I_h, S \subseteq E \setminus I_h$ such that $|R| \geq |S|$ and $I_h \setminus R \cup S \in \mathcal{I}$. Assume further that $I_h + s \notin \mathcal{I}$ for all $s \in S$. Then $I_{h+1} \setminus R \cup S \in \mathcal{I}$ as well: if I_{h+1} is derived by deletion of an element, this follows immediately. Now assume that $I_{h+1} = I_h + i$. Since $I_h + i \in \mathcal{I}$ and $I_h \setminus R \cup S \in \mathcal{I}$, by matroid augmentation property there exists some $j \in (I_h + i) \setminus (I_h \setminus R \cup S) = R + i$ with $I_h \setminus R \cup S + j \in \mathcal{I}$. If $j = i$ we are done, otherwise we get a contradiction: suppose that $I_h \setminus (R - j) \cup S \in \mathcal{I}$. Then we can apply the matroid augmentation property to I_h to find some $s \in S$ with $I_h + s \in \mathcal{I}$.

Now consider a subsequence $I_h, I_{h+1}, \ldots, I_g$ where no element of B_0 is added. Then if $I_h \setminus R \cup S \in \mathcal{I}$ it follows that $I_g \setminus R \cup S \in \mathcal{I}$. Notice that as we have shown earlier, for every $R \subseteq A \subseteq C$ with $|R| \geq \epsilon|A|$ there exists some $S_1 \subseteq B_0$ such that $|S_1| \geq \epsilon^2|B_0|$ and $I_1 \setminus R \cup S_1 \in \mathcal{I}$. Let I_h be the first time that an element of B_0 is inserted into I_M. Then by previous arguments $I_{h-1} \setminus R \cup S_1 \in \mathcal{I}$. Since I_h extends I_{h-1} by only one element, by matroid augmentation property $I_h \setminus R \cup S_2 \in \mathcal{I}$ for some $S_2 \subseteq S_1$ with $|S_2| = |S_1| - 1$. We can repeat this argument and since only $|I_k \cap B_0|$ many times an element from B_0 is added, we will finally obtain a set S_k with $|S_k| = |S_1| - |B_0 \cap I_k| \geq \epsilon^2|B_0| - |B_0 \cap I_k|$ and $I_k \setminus R \cup S_k \in \mathcal{I}$; thus proving the lemma. ∎

LEMMA 5.6. *At all times, I_M and I_P are disjoint, $I_M \in \mathcal{I}$, and $b \cdot I_P \in \mathcal{P}$.*

Proof. The only modifications to I_M are $I_M \leftarrow C \cup I''_M$ through recursion, where I''_M is independent in the contracted matroid $\mathcal{M}/C$ and greedy additions of elements before terminating. Both of these operations clearly maintain $I_M \in \mathcal{I}$.

For I_P, we will argue the stronger statement that at all times $b \cdot (I_P \cup A_I) \in \mathcal{P}$, which also implies that the operation of adding elements from A_I to I_P will maintain $I_P \in \mathcal{P}$. The property that $b \cdot (I_P \cup A_I) \in \mathcal{P}$ is by definition of the algorithm maintained when adding elements to A_I. Consider now the operation $I_P \leftarrow (B \setminus I''_M) \cup I''_P$ performed by the recursion, where each element $i \in I''_P$ satisfies $f(i \mid b \cdot (I''_P \cup A \cup B - i)) \geq b$. We assume that before the recursive call we have $b \cdot (I_P \cup A_I) \in \mathcal{P}$ and, in particular, $b \cdot (B \cup A_I) \in \mathcal{P}$. Thus, because of the marginal values of all elements in I''_P, it follows immediately that $b \cdot (A_I \cup B \cup I''_P) \in \mathcal{P}$ and, in particular, $b \cdot (A_I \cup (B \setminus I''_M) \cup I''_P) \in \mathcal{P}$. Note that $(B \setminus I''_M) \cup I''_P$ is equal to I_P after the recursion. Since these are the only places where I_P is changed, this completes the proof. ∎

LEMMA 5.7. *The input $E', B'_0, I'_M, I'_P, \mathcal{M}', \mathcal{P}'$ created for the recursion is feasible. More concretely,*

1. *$B'_0, I'_M, I'_P \subseteq E'$ are disjoint,*
2. *$I'_M \in \mathcal{I}'$, and*
3. *$b \cdot I'_P \in \mathcal{P}'$.*

Proof. The only non-obvious statement is $b \cdot I'_P \in \mathcal{P}'$. Here, notice that $f'(X) = f(X \mid b \cdot (A \cup B))$ and for each $i \in I'_P$ we have $f(i \mid b \cdot (A \cup I_P - i) \geq b$, since $i \notin B$. Starting with $X = \emptyset$ and adding each element of I'_P one at a time, it is easy to see that the marginal values $f'(i \mid b \cdot X)$ are always least b. ∎

LEMMA 5.8. *For all $i \in A \cup B \setminus A_I$ we have $f(i \mid b \cdot (A \cup B - i)) < b$.*

Proof. Consider the time B was last updated. By definition we have $f(i \mid b \cdot (A \cup I_P - i)) < b$ if and only if $i \in B$ for all $i \in I_P$. Further, for every $i \in A \setminus A_I$ we have $f(i \mid b \cdot (A \cup I_P - i)) \leq f(i \mid b \cdot I_P) < b$.

Let $X = \{i \in A \cup I_P : f(i \mid b \cdot (A \cup I_P - i) < b\}$. Then by the previous observations, $A \cup B \setminus A_I \subseteq X \subseteq A \cup B$. By Lemma 5.2 it follows that $f(i \mid b \cdot (A \cup B - i)) \leq f(i \mid b \cdot (X - i)) < b$ for all $i \in A \cup B \setminus A_I$. ∎

We will now prove that a recursion significantly decreases the number of blocking elements.

LEMMA 5.9. *Let B' be the set of blocking elements after a recursion has returned and assume that the algorithm did not immediately terminate. Denote by B the blocking elements before the recursion. Then $B' \subseteq B$. Moreover, $|B'| \leq (1 - \epsilon^2)|B|$.*

Proof. Let $i \in I'_P \setminus B$, where again the prime denotes the state after the recursion. Then $f(i \mid b \cdot (A \cup I'_P - i)) \geq b$ (using the definition of the submodular function f' of the recursion). Therefore, $i \notin B'$. Since $|I'_M \cap B| \geq \epsilon^2 |B|$ and such elements will not appear in I'_P it follows immediately that $|B'| \leq (1 - \epsilon^2)|B|$. ∎

LEMMA 5.10. *At all times we have $|B| > (1 - 2\epsilon)|A|$*

Proof. Recall that for all $i \in A \cup B$ we have $f(i \mid b \cdot (A \cup B - i)) < b$ unless $i \in A_I$, see Lemma 5.8. Thus, from Lemma 5.3 it follows that $f((A \setminus A_I) \cup B) < b|A \cup B|$. Furthermore, $2b \cdot A \in \mathcal{P}$ and $|A_I| < \epsilon |A|$, which implies that $f((A \setminus A_I) \cup B) \geq f(A \setminus A_I) \geq 2b|A \setminus A_I| \geq (2 - 2\epsilon)b|A|$. Putting both inequalities together, we get $|A \cup B| > (2 - 2\epsilon)|A|$, which simplifies to $|B| > (1 - 2\epsilon)|A|$. ∎

Termination with failure. In the case that our algorithm returns failure, we need to prove that there does not exist a (slightly stronger) solution. This proof is in the form of a certificate. We will start with some intuition and background on simpler certificates that do not work.

First, it is unclear how one would generalize the configuration LP beyond the partial matroid generalization of Davies et al. [15], since it heavily relies on the matching structure of small resources. But even worse, we show that already in a special case for which the configuration LP is still meaningful, its integrality gap is large; hence it is not helpful. Consider the following instance (which appeared already in [7]). We have a first set E of m

players, and a set S of $m-1$ resources. For each player $i \in E$, there is an additional set E_i of m players and an additional set S_i of $m+1$ resources. Each player $i \in E$ has valuation m for all the resources in S, valuation 1 for all the resources in E_i, and valuation 0 for the remaining resources. For any $i \in E$, each player in E_i has valuation m for the resources in S_i, and valuation 0 for other resources. It was showed in [7] that in this example the configuration LP will give an optimal objective value of m for this instance, while it is clear that the integral optimum is at most 1. Indeed, there are only $m-1$ resources in S, hence at least one player i from E will need to take resources from S_i. But S_i contains only $m+1$ resources while m players in E_i need to take one resource each from the corresponding set S_i. Interestingly, this example can be captured by our polymatroid variant. We have universe E and the uniform matroid of rank $|E|-1$ (i.e. $r(X) = |X|$ for any $X \neq E$, and $r(E) = |E|-1$), and $f(X) = |X|$, which models that a set of $|X|$ players in E_1 can be assigned a maximum of $|X|$ resources from the set $\bigcup_{i \in E} S_i$. In the matroid problem, the players in $\bigcup_{i \in E} E_i$ only appear implicitly. One may also derive f by defining a natural polymatroid that assigns the items in $\bigcup_{i \in E} S_i$ and from which we then contract the players of $\bigcup_{i \in E} E_i$. As stated above, the configuration LP does not give us a good lower bound (or certificate of infeasibility) in this case. Another way to find a certificate would be, in the spirit of matroid partition, to try to find a set Z, for which $r(Z) + f(Z)/b < |Z|$, where r is the rank function of the matroid and f the submodular function corresponding to the polymatroid. Although this would prove infeasibility of a solution of value b, it is not sufficient as seen in the example above. As previously detailed, it is clear that no solution of value more than 1 exists. On the other hand, $r(Z)+f(Z)/|Z| \geq r(Z) = |Z|$ for all $Z \neq E$ and $r(E)+f(E)/|E| = |E|-1+1 \geq |E|$. Hence, the certificate above is also not sufficient to rule out a solution of value $|E| = m$.

To overcome this issue, we develop a new idea, which first introduce in the following simplified way. Let $Z_2 \subseteq E$ with $f(Z_2) \leq b \cdot |Z_2|$, in the example above we can take $Z_2 = \emptyset$. Further, let $Z_1 \supseteq Z_2$ have a not too large rank of $r(Z_1) < |Z_1| - |Z_2|/2$ and small marginal values for each element, that is, $f(i \mid Z_2) < b$ for all $i \in Z_1 \setminus Z_2$. In the example above, take $Z_1 = E$. We claim that this constitutes a proof that no solution of value $3b$ exists. Suppose that for $I_P \subseteq Z_1$ we have that $3b \cdot I_P$ is in the polymatroid. Then

$$2b \cdot |I_P| \leq 3b \cdot |I_P| - \sum_{i \in I_P \setminus Z_2} f(i \mid Z_2) \leq f(Z_2) \leq b \cdot |Z_2| \ ,$$

where we use that $3b \cdot |I_P| \leq f(I_P) \leq f(Z_2) + \sum_{i \in I_P \setminus Z_2} f(i \mid Z_2)$. It follows that $|I_P| \leq |Z_2|/2$, but because of its rank the matroid cannot cover all remaining elements. In the formal definition of the certificate we need a more complicated variant. This is mainly for efficiency reasons: in order to achieve polynomial running time, the algorithm uses some parameter ϵ and this forces us to work with weaker conditions.

Definition 5.1. A *certificate of infeasibility* consists of two (possibly empty) sets $Z_2 \subseteq Z_1 \subseteq E \setminus B_0$ with

1. $r(B_0 \mid Z_1) < 2\epsilon|B_0|$,
2. $r(Z_1) \leq |Z_1| - (0.5 - 2\epsilon)|Z_2| + \epsilon|B_0|$,
3. $f(i \mid b \cdot (Z_2 - i)) < b$ for at least a $(1-\epsilon)$ fraction of elements in Z_2,
4. $f(i \mid 2b \cdot Z_2) < 2b$ for all $i \in Z_1 \setminus Z_2$

We briefly compare this definition to the simpler variant before. First, Property 3 implies that $f(Z_2') \leq b \cdot |Z_2|$ for some large subset $Z_2' \subseteq Z_2$ by Lemma 5.3; hence very similar of the simpler variant. Property 2 is weaker than the simpler variant by the additive term of $\epsilon|B_0|$. This, however, is needed so that we arrive at a certificate also when the algorithm cannot make much process. To compensate for it we introduce Property 1. This property implies that if we wanted to cover $3\epsilon|B_0|$ many elements of B_0 with the matroid, then the number of elements the matroid covers in Z_1 must be at most $r(Z_1) - \epsilon|B_0|$, i.e., canceling out the increase in the bound of Property 2. Together we obtain a certificate that shows that not many elements of B_0 can be covered by the matroid, which is formalized below.

Lemma 5.11. *If there exists a certificate of infeasibility then there cannot be two sets $I_M^* \cup I_P^* \supseteq E \setminus B_0$ with $I_M^* \in \mathcal{I}$, $\alpha b \cdot I_P^* \in \mathcal{P}$, and $|B_0 \cap I_M^*| \geq 3\epsilon|B_0|$, where $\alpha = 4 + \mathcal{O}(\epsilon)$.*

Proof. Let $I_M^* \cup I_P^* \supseteq E \setminus B_0$ with $I_M^* \in \mathcal{I}$, $\alpha b \cdot I_P^* \in \mathcal{P}$. Let $Z_2' \subseteq Z_2$ be the elements with $f(i \mid b \cdot (Z_2 - i)) \geq b$. Then by (3) we have $|Z_2'| \leq \epsilon \cdot |Z_2|$ and from Lemma 5.3 it follows that $f(Z_2 \setminus Z_2') \leq b \cdot |Z_2|$. We first bound

$$\begin{aligned}
\alpha b|I_P^* \cap (Z_1 \setminus Z_2')| &\leq f(I_P^* \cap (Z_1 \setminus Z_2')) \\
&\leq f(Z_2 \setminus Z_2') + f(I_P^* \cap (Z_1 \setminus Z_2) \mid Z_2 \setminus Z_2') \\
&\leq f(Z_2 \setminus Z_2') + f(I_P^* \cap (Z_1 \setminus Z_2) \mid 2b \cdot (Z_2 \setminus Z_2')) \\
&\leq f(Z_2 \setminus Z_2') + 2b|Z_2'| + f(I_P^* \cap (Z_1 \setminus Z_2) \mid 2b \cdot Z_2) \\
&\leq f(Z_2 \setminus Z_2') + 2b|Z_2'| + \sum_{i \in I_P^* \cap (Z_1 \setminus Z_2)} f(i \mid 2b \cdot Z_2) \\
&\leq f(Z_2 \setminus Z_2') + 2b|Z_2'| + \sum_{i \in I_P^* \cap (Z_1 \setminus Z_2')} f(i \mid 2b \cdot Z_2) \ .
\end{aligned}$$

From this it follows that

$$b(\alpha - 2)|(Z_1 \setminus Z_2') \cap I_P^*| \leq \sum_{i \in I_P^* \cap (Z_1 \setminus Z_2')} (\alpha b - f(i \mid 2b \cdot (Z_2 - i))) \leq 2\epsilon b|Z_2| + f(Z_2 \setminus Z_2') \leq (1 + 2\epsilon)b|Z_2| \ .$$

Consequently, $|(Z_1 \setminus Z_2') \cap I_P^*| \leq (1 + 2\epsilon)/(\alpha - 2) \cdot |Z_2|$. Thus,

$$|Z_1 \cap I_M^*| \geq |Z_1| - |(Z_1 \setminus Z_2') \cap I_P^*| - |Z_2'| \geq |Z_1| - \frac{1 + 2\epsilon + (\alpha - 2)\epsilon}{\alpha - 2}|Z_2| \geq r(Z_1) - \epsilon|B_0| \ ,$$

where the last inequality holds because of Property (2) for $\alpha = 4 + \mathcal{O}(\epsilon)$ and ϵ sufficiently small. Thus,

$$\begin{aligned}
|B_0 \cap I_M^*| \leq r(B_0 \mid I_M^* \cap Z_1) = r(B_0 \cup (I_M^* \cap Z_1)) - r(I_M^* \cap Z_1) \\
\leq r(B_0 \cup Z_1) - r(Z_1) + \epsilon|B_0| = r(B_0 \mid Z_1) + \epsilon|B_0| < 3\epsilon|B_0| \ ,
\end{aligned}$$

which concludes the proof. ∎

Next we will prove that whenever the algorithm returns failure, there exists such a certificate.

LEMMA 5.12. *If $|B| < \epsilon|B_0|$ then there exists a certificate that proves infeasibility.*

Proof. We set $Z_2 = A \cup B$ and $Z_1 = C \cup B$. Then by Lemma 5.4 we have

$$r(B_0 \mid Z_1) \leq r(B_0 \mid C) < 2\epsilon|B_0| \ .$$

Further,

$$\begin{aligned}
r(Z_1) \leq r(C) + r(B) = |C| + \epsilon|B_0| \leq |Z_1| + \epsilon|B_0| - 0.5|B| - 0.5|B| \\
\leq |Z_1| + \epsilon|B_0| - 0.5|B| - (0.5 - \epsilon)|A| \leq |C_1| - (0.5 - 2\epsilon)|Z_2| + 2\epsilon|B_0| \ .
\end{aligned}$$

Here we use that $|B| \geq (1 - 2\epsilon)|A|$, see Lemma 5.10. It follows from Lemma 5.3 and Lemma 5.1 that $f(i \mid b \cdot (Z_2 - i)) \leq f(i \mid b \cdot (A \cup B - i)) < 2b$ for each $i \in C_2 \setminus A_I$ and $|A_I| \leq \epsilon|A| \leq \epsilon|Z_2|$ since otherwise we would have terminated successfully.

Finally, $f(i \mid 2b \cdot C_2) \leq f(i \mid 2b \cdot A) < 2b$ for all $i \in C \setminus A$ follows immediately from the definition of C. ∎

LEMMA 5.13. *When a recursive call returns failure, then there exists a certificate that proves infeasibility.*

Proof. Assume that a recursive call has failed. Let Z_1', Z_2' be the certificate returned by it. We set $Z_1 = Z_1' \cup C \cup B$ and $Z_2 = Z_2' \cup A \cup B$. Then by Lemma 5.4,

$$r(B_0 \mid Z_1) \leq r(B_0 \mid C) < 2\epsilon|B_0| \ .$$

Further,

$$\begin{aligned} r(Z_1) &= r(Z_1' \cup C \cup B) \\ &= r(Z_1' \mid C) + r(C) + r(B \mid Z_1' \cup C) \\ &\leq r'(Z_1') + |C| + r'(B \mid Z_1') \\ &\leq |Z_1'| - (0.5 - 2\epsilon)|Z_2'| + 2\epsilon|B| + |C| + \epsilon|B_0| \\ &= |Z_1| - (0.5 - 2\epsilon)|B| - 0.5|B| - (0.5 - 2\epsilon)|Z_2'| + \epsilon|B_0| \\ &\leq |Z_1| - (0.5 - 2\epsilon)|Z_2| + \epsilon|B_0| \ . \end{aligned}$$

Moreover, for all $i \in C_2 \setminus A_I$ it holds that $f(i \mid b \cdot (Z_2 - i)) \leq f(i \mid b \cdot (A \cup B - i)) < 2b$ Similarly, for a $1 - \epsilon$ fraction of Z_2' it holds that $f(i \mid b \cdot (Z_2 - i)) \leq f(i \mid b \cdot (Z_2' - i)) < b$. Since $|A_I| < \epsilon|A| \leq \epsilon|A \cup B|$ Property (3) is satisfied. Likewise $f(i \mid 2b \cdot Z_2) \leq f(i \mid 2b \cdot Z_2') < 2b$ for all $i \in Z_1' \setminus Z_2$ and $f(i \mid 2b \cdot Z_2) \leq f(i \mid A) < 2b$ for all $i \in C \setminus Z_2$. Thus, also the last property holds. □

5.2 Running time We are going to bound only the number of nodes in the recursion tree. It is clear that the overhead of operations outside the recursive call is polynomially bounded. To this end, we focus on the sets B_0 and B. Initially, the set B created in the algorithm will have size at most n. Then with every recursive call it decreases by a factor of $(1 - \epsilon^2)$, see Lemma 5.9, but never below $\epsilon|B_0|$ (else, the algorithm terminates immediately).

For a fixed instance, let $T(k)$ be the maximum number of nodes in the recursion tree of the algorithm over all inputs B_0, X, Y where $|B_0| \geq 1/(1 - \epsilon^2)^k$. Then $T(k)$ is monotone decreasing, i.e., $T(k') \leq T(k)$ for $k' \geq k$, simply because the set over which the maximum is taken is smaller. Let $\ell = \lceil \log_{1/(1-\epsilon^2)^k}(n) \rceil$. Then,

$$T(k) \leq 1 + \sum_{i=k+1}^{\ell} T(i) \quad \text{and} \quad T(\ell) = 1 \ .$$

Here, the sum starts at $k+1$, since the smallest size of B_0' of the recursion satisfies $|B_0'| \geq (1+\epsilon)|B_0| \geq |B_0|/(1-\epsilon^2)$. The numbers $T(\ell), T(\ell-1), T(\ell-2), \ldots, T(1)$ are similar to the Fibonacci series (except for the addition of 1 for the current node) and it can easily be shown by induction that $T(k) \leq 2^{\ell-k} \leq 2^{\ell} \leq n^{\mathcal{O}_\epsilon(1)}$ for all k.

6 Final remarks

For the two notorious open problems in scheduling theory, we prove Makespan to be at least as difficult as SantaClaus; more precisely, a better-than-2 approximation for Makespan would imply an $\mathcal{O}(1)$-approximation for SantaClaus. In the two-value case both problems appear equivalent w.r.t. approximability. The obvious open question is whether there is also a Makespan-to-SantaClaus reduction (for restricted assignment or the general case). Here we note that for restricted assignment Makespan, all efforts to refine the local search method in order to give a better-than-2 approximation have failed so far. Also with our new reduction techniques it seems that it would require additional ideas to handle this problem. By the reductions, our local search method generalizes all previously known polynomial-time local search results for the two problems (up to constants) and yields the first approximation algorithm for a new matroid SantaClaus variant, where items are allocated to the basis of a (poly-)matroid. We hope that this makes it clearer where the power and limitations of the method are.

Finally, we comment on an alternative matroid scheduling variant with matroid constraints on the *items* allocated to a specific machine/player. In the *machine-matroid* Makespan problem, each machine would be given a matroid on the jobs. All jobs must be assigned such that each machine receives an independent set of its matroid. The player-matroid SantaClaus can be defined similarly.

Kawase et al. [23] consider such matroid partition problems for various objective functions showing complexity results. Further, two special-matroid examples for Makespan have been studied, namely, bag-constrained scheduling [14,17] (single partition matroid) and scheduling with capacity constraints [11] (uniform matroids). The approximability lower bound $\Omega((\log n)^{1/4})$ by [14] holds for the restricted assignment setting and even translates to an inapproximability bound for machine-matroid Makespan for identical machines with machine-dependent matroids. We are not aware of any similarly strong lower bounds for the SantaClaus variant.

References

[1] Chidambaram Annamalai. Lazy local search meets machine scheduling. *SIAM Journal on Computing*, 48(5):1503–1543, 2019.

[2] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. *ACM Transactions on Algorithms*, 13(3):1–28, 2017.

[3] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24:1–24:9, 2012.

[4] Yossi Azar, Jaya Prakash Champati, and Ben Liang. 2-approximation algorithm for a generalization of scheduling on unrelated parallel machines. *Information Processing Letters*, 139:39–43, 2018.

[5] Etienne Bamas and Lars Rohwedder. Better trees for santa claus. In *Proceedings of STOC*, 2023.

[6] Nikhil Bansal. Scheduling: Open problems old and new, 2017. Invited Presentation at MAPSP 2017, Kloster Seeon, Germany.

[7] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *Proceedings of STOC*, pages 31–40, 2006.

[8] Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *ACM SIGecom Exchanges*, 5(3):11–18, 2005.

[9] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *Proceedings of FOCS*, pages 107–116, 2009.

[10] Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. On (1,ϵ)-restricted assignment makespan minimization. In *Proceedings of SODA*, pages 1087–1101, 2014.

[11] Lin Chen, Klaus Jansen, Wenchang Luo, and Guochuan Zhang. An efficient PTAS for parallel machine scheduling with capacity constraints. In *Proceedings of COCOA*, pages 608–623. Springer, 2016.

[12] Siu-Wing Cheng and Yuchen Mao. Restricted max-min fair allocation. In *Proceedings of ICALP*, volume 107, pages 37:1–37:13, 2018.

[13] Siu-Wing Cheng and Yuchen Mao. Restricted max-min allocation: Approximation and integrality gap. In *Proceedings of ICALP*, pages 38:1–38:13, 2019.

[14] Syamantak Das and Andreas Wiese. On minimizing the makespan when some jobs cannot be assigned on the same machine. In *Proceedings of ESA*, pages 31:1–31:14, 2017.

[15] Sami Davies, Thomas Rothvoss, and Yihao Zhang. A tale of santa claus, hypergraphs and matroids. In *Proceedings of SODA*, pages 2748–2757, 2020.

[16] Uriel Feige. On allocations that maximize fairness. In *Proceedings of SODA*, pages 287–293, 2008.

[17] Kilian Grage, Klaus Jansen, and Kim-Manuel Klein. An EPTAS for machine scheduling with bag-constraints. In *Proceedings of SPAA*, pages 135–144, 2019.

[18] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the lovász local lemma. *Journal of the ACM*, 58(6):28:1–28:28, 2011.

[19] Penny Haxell and Tibor Szabó. Improved integrality gap in max-min allocation: or topology at the north pole. In *Proceedings of SODA*, pages 2875–2897, 2023.

[20] Penny E Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.

[21] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.

[22] Klaus Jansen and Lars Rohwedder. A quasi-polynomial approximation for the restricted assignment problem. *SIAM Journal on Computing*, 49(6):1083–1108, 2020.

[23] Yasushi Kawase, Kei Kimura, Kazuhisa Makino, and Hanna Sumita. Optimal matroid partitioning problems. *Algorithmica*, 83(6):1653–1676, 2021.

[24] Donald Ervin Knuth. *Matroid partitioning.* Computer Science Department, Stanford University, 1973.

[25] Jan Karel Lenstra, David B Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1):259–271, 1990.

[26] Lukáš Poláček and Ola Svensson. Quasi-polynomial local search for restricted max-min fair allocation. *ACM Transactions on Algorithms*, 12(2):1–13, 2015.

[27] Alexander Schrijver. *Combinatorial Optimization − Polyhedra and Efficiency.* Springer, 2003.

[28] Petra Schuurman and Gerhard J. Woeginger. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, 2(5):203–213, 1999.

[29] Evgeny V. Shchepin and Nodari Vakhania. An optimal rounding gives a better approximation for scheduling unrelated machines. *Oper. Res. Lett.*, 33(2):127–133, 2005.

[30] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62:461–474, 1993.

[31] Ola Svensson. Santa claus schedules jobs on unrelated machines. *SIAM Journal on Computing*, 41(5):1318, 2012.

[32] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

[33] Gerhard J. Woeginger. Open problems in the theory of scheduling. *Bull. EATCS*, 76:67–83, 2002.

A Combining polymatroids of the same value

PROPOSITION 1.1. *For any $\alpha \geq 1$, if there exists a polynomial-time α-approximation algorithm for restricted job-matroid* MAKESPAN *(resource-matroid* SANTACLAUS*) with h jobs (resources), then there exists a polynomial-time α-approximation algorithm for restricted job-matroid* MAKESPAN *(resource-matroid* SANTACLAUS*) with p_j resp. $v_j \in \{w_1, \dots, w_h\}$ and $w_1, \dots, w_h \geq 0$.*

Proof. In the following, we use the notation $[h] := \{1, \dots, h\}$. Let I be an instance of the restricted job-matroid MAKESPAN problem with machines E and h distinct processing times $p_1, \dots, p_h$. Let J_ℓ, $\ell \in [h]$, denote the set of jobs with processing times p_ℓ. Further, let $\mathcal{P}_j^\ell$ with $\ell \in [h]$ and $j \in J_\ell$ denote the corresponding polymatroids over E and let f_j^ℓ be the associated submodular function.

We construct an instance I' of the restricted job-matroid MAKESPAN problem with h jobs by using the same set of machines E and creating the polymatroids $\mathcal{P}_\ell$ for $\ell \in [h]$ with the montone submodular function $f_\ell(S) = \sum_{j \in J_\ell} f_j^\ell(S)$ for every subset $S \subseteq E$. Note that $\mathcal{P}_\ell = \sum_{j \in J_\ell} \mathcal{P}_j^\ell$ [27]. For $\ell \in [h]$, the goal in instance I' is to find vectors $x_\ell \in \mathcal{B}(\mathcal{P}_\ell)$ such that $\max_{e \in E} \sum_{\ell \in [h]} p_\ell \cdot x_\ell(e)$ is minimized. We prove that this reduction preserves the approximation factor.

Consider a solution of instance I that selects the bases x_j^ℓ for job $j \in J_\ell$ with $\ell \in [h]$ and consider the vectors x'_ℓ with $x'_\ell(e) = \sum_{j \in J_\ell} x_j^\ell(e)$ for all $e \in E$. Using again the fact that $\mathcal{P}_\ell = \sum_{j \in J_\ell} \mathcal{P}_j^\ell$ we have $x'_\ell \in \mathcal{P}_\ell$ for all $\ell \in [h]$. In particular, $x'_\ell(E) = \sum_{e \in E} \sum_{j \in J_\ell} x_j^\ell(e) = \sum_{j \in J_\ell} f_j^\ell(E) = f_\ell(E)$, so x'_ℓ is a basis of $\mathcal{P}_\ell$. Thus, $(x'_1, \dots, x'_h)$ is a feasible solution for instance I'. Furthermore,

$$\mathrm{OPT}(I') \leq \max_{e \in E} \sum_{\ell \in [h]} x'_\ell(e) \cdot p_\ell = \max_{e \in E} \sum_{\ell \in [h]} \sum_{j \in J_\ell} x_j^\ell(e) \cdot p_\ell = \mathrm{OPT}(I).$$

Consider some solution $(y'_1, \dots, y'_h)$ to instance I', i.e., $y'_\ell \in \mathcal{B}(\mathcal{P}_\ell)$ and $f_\ell(E) = y'_\ell(E)$ for all $\ell \in [h]$. We construct a solution to I by decomposing each y'_ℓ, $\ell \in [h]$, into bases $y_j^\ell \in \mathcal{B}(\mathcal{P}_j^\ell)$ such that $y'_\ell(e) = \sum_{j \in J_\ell} y_j^\ell(e)$ holds for all $e \in E$. As $\mathrm{OPT}(I') \leq \mathrm{OPT}(I)$, this implies that the reduction preserves the approximation factor. If such decomposition would not exist for some $\ell \in [h]$, then, by construction of the submodular function f_ℓ, we would arrive at a contradiction to $f_\ell(E) = y'_\ell(E)$.

To find the decomposition for an $\ell \in [h]$ in polynomial time, consider the polymatroids $\hat{\mathcal{P}}_j^\ell$ which are just copies of the original polymatroids $\mathcal{P}_j^\ell$ on pairwise disjoint copies $\hat{E}_j$ of the ground set E. For each $\ell \in [h]$, we decompose the solution y'_ℓ of instance I' into bases of the copy polymatroids, which then implies a decomposition into bases of the original polymatroids. For each $e \in E$, let C_e denote the set of copies of e introduced by the ground set copies. We want to find a basis $\hat{y}_j^\ell$ for every $j \in J_\ell$ such that $\sum_{\hat{e} \in C_e} \hat{y}_j^\ell(\hat{e}) = y'_\ell(e)$ holds for all $e \in E$ and $\ell \in [h]$. For an element $e \in E$ and $\ell \in [h]$, consider the polymatroid $\mathcal{X}_e^\ell$ on the ground set C_e implied by bases $\mathcal{B}(\mathcal{X}_e^\ell) = \{x \in \mathbb{Z}_{\geq 0}^{C_e} : x(C_e) = y'_\ell(e)\}$ and let $\mathcal{X}^\ell$ denote the union of these polymatroids. Furthermore, let $\hat{\mathcal{P}}^\ell$ denote the union of the polymatroids $\hat{\mathcal{P}}_j^\ell$. The largest element in the intersection of $\mathcal{X}^\ell$ and $\hat{\mathcal{P}}^\ell$ gives us the decomposition. We can compute the largest element in the intersection in polynomial time using algorithms for polymatroid intersection (cf. e.g. [27, Chapter 41]).

The statement for resource-matroid SANTACLAUS can be shown with the same reduction and proof; only the inequality $\mathrm{OPT}(I') \leq \mathrm{OPT}(I)$ trivially changes to $\mathrm{OPT}(I') \geq \mathrm{OPT}(I)$. ∎

B Santa Claus with polynomially many configurations

LEMMA 2.2. *For every $\epsilon > 0$ and a given instance I of* SANTACLAUS *with $\mathrm{OPT}(I) \geq 1$, we can construct a rounded instance I' with a collection of configurations $\mathcal{C}$ such that the number of configurations for each player is polynomial in the input size of I and $\mathrm{OPT}_{\mathcal{C}}(I') \geq 1/(1+\epsilon)$.*

Further, every solution for I' of objective value T is a solution for I with objective value at least T.

Proof. Let $\epsilon > 0$ be a sufficiently small constant and $\kappa = \lceil 1/\epsilon^3 \rceil$. Given a SANTACLAUS instance I with the set P of m players, the set R of n resources and $\mathrm{OPT}(I) \geq 1$, we construct the SANTACLAUS instance I' by executing the following steps.

1. Use the same set of players and resources as in I.

2. Round all resource values v_{ij} down to the closest power of $1/(1+\epsilon)$. That is, we round $1/(1+\epsilon)^{\ell-1} \geq v_{ij} \geq 1/(1+\epsilon)^{\ell}$ to $\bar{v}_{ij} = 1/(1+\epsilon)^{\ell}$. If $v_{ij} \geq 1$, then we set $\bar{v}_{ij} = 1$. Furthermore, we round all v_{ij} with $v_{ij} < 1/((1+\epsilon)n)$ to zero. In summary, each $\bar{v} \in \mathcal{T}$ is either a power of $1/(1+\epsilon)$ of value at least $1/((1+\epsilon)n)$ or 0.

Next, we construct the configurations for the rounded instance I' by executing the following steps. Since these steps will reduce the number of possible configurations per player to a polynomial, an algorithm creating the configurations can just compute them via enumeration.

3. For each player $i \in P$, we create the set $\mathcal{C}_i$ of configurations and restrict the set to configurations c such that, for every value type $\bar{v} \in \mathcal{T}$, either $c(\bar{v}) = 0$, $c(\bar{v}) = \lceil (1+\epsilon)^{\ell} \rceil$ or $c(\bar{v}) = \lfloor (1+\epsilon)^{\ell} \rfloor$ for some $\ell \in \mathbb{N}_0$ with $(1+\epsilon)^{\ell} \leq n$.

4. Let $\bar{v}_1 \geq \ldots \geq \bar{v}_\tau$ be the rounded value types in $\mathcal{T}$. We partition $\mathcal{T}$ into κ value classes $\mathcal{T}_1, \ldots, \mathcal{T}_\kappa$ where $\mathcal{T}_\ell = \{\bar{v}_{\ell+s\cdot\kappa} : s = 0, 1, \ldots\}$. We further restrict the set of configurations $\mathcal{C}_i$ for a player $i \in P$ to configurations c which satisfy for every $1 \leq \ell \leq \kappa$ and for every $\bar{v}, \bar{v}' \in \mathcal{T}_\ell$ with $\bar{v} > \bar{v}'$ that either $c(\bar{v}) < c(\bar{v}')$ or $c(\bar{v}) = 0$ or $c(\bar{v}') = 0$. That is, the function values of value types $\bar{v} \in \mathcal{T}_\ell$ of one value class that actually occur in a configuration (i.e., have $c(\bar{v}) > 0$) increase with decreasing value $\bar{v} \in \mathcal{T}_\ell$.

We first argue that, for each player i, the number of configurations in $\mathcal{C}_i$ is polynomial in the input size. Because of the second step, the number of value types in I' is in $\mathcal{O}_\epsilon(\log n)$. By the third step, the number of distinct function values $c(\bar{v})$ over all configurations $c \in \mathcal{C}_i$ and all value types $\bar{v} \in \mathcal{T}$ is in $\mathcal{O}_\epsilon(\log n)$ as well.

By step 4, we can represent the entries of $c \in \mathcal{C}_i$ which correspond to the same value class $\mathcal{T}_\ell$ in terms of a vector with $\mathcal{O}_\epsilon(\log n)$ entries such that all non-zero entries strictly increase in value (each entry of the vector corresponds to a value type $\bar{v} \in \mathcal{T}_\ell$, in decreasing order, and the entry values represent the corresponding function values $c(\bar{v})$). We can represent such a vector by the set of entries that have a non-zero value and by the set of non-zero values that occur in the vector. Since there are at most $2^{\mathcal{O}_\epsilon(\log n)}$ different sets of non-zero values that can occur in the vector and at most $2^{\mathcal{O}_\epsilon(\log n)}$ different sets of non-zero entries, the number of such vectors is $2^{\mathcal{O}_\epsilon(\log n)} \cdot 2^{\mathcal{O}_\epsilon(\log n)} \subseteq n^{\mathcal{O}_\epsilon(1)}$. Since the total number of value classes is constant, and there is a simple rule to compose the vectors of every class to a vector for all value types $\mathcal{T}$, the total number of vectors which represent every valid configuration is polynomial in the size of I.

Next, we prove the approximation factor. Let I' denote the rounded instance constructed by the first two steps. Furthermore, let $\mathcal{C}'$ denote the collection of configurations that is created by only executing the third step of the construction and let $\mathcal{C}$ denote the final collection of configurations. We separately prove $\mathrm{OPT}_{\mathcal{C}'}(I') \geq 1/(1+\epsilon)^3$ and $\mathrm{OPT}_{\mathcal{C}}(I') \geq \mathrm{OPT}_{\mathcal{C}'}(I')/(1+\epsilon)$. Together, these inequalities imply $\mathrm{OPT}_{\mathcal{C}}(I') \geq 1/(1+\epsilon)^4$. Then, for any sufficiently small $\epsilon' > 0$, we can choose $\epsilon = \epsilon'/5$ and conclude $\mathrm{OPT}_{\mathcal{C}}(I') \geq 1/(1+\epsilon')$.

We first show $\mathrm{OPT}_{\mathcal{C}'}(I') \geq 1/(1+\epsilon)^3$. Consider some optimal solution for I. For a player i, let A_i denote the resources that are assigned to i in the optimal solution. Clearly $v(A_i) \geq 1$. Discarding all resources in A_i with value smaller than $1/((1+\epsilon)n)$ reduces the value of A_i by a factor of at most $1+\epsilon$. Rounding the remaining resource values down to powers of $1+\epsilon$ reduces the value by another factor of $1+\epsilon$. To make sure that the remaining resources in A_i with their rounded values match a configuration in $\mathcal{C}'_i$, we might have to remove a $1+\epsilon$ fraction of the resources for each value type from A_i. This reduces the value of A_i by another factor of $1+\epsilon$. The remaining value is at least $1/(1+\epsilon)^3$. By doing this for every player i, we obtain a solution to I' that matches $\mathcal{C}'$ with an objective value of at least $1/(1+\epsilon)^3$, which implies $\mathrm{OPT}_{\mathcal{C}'}(I') \geq 1/(1+\epsilon)^3$.

Finally, we prove $\mathrm{OPT}_{\mathcal{C}}(I') \geq \mathrm{OPT}_{\mathcal{C}'}(I')/(1+\epsilon)$. To that end, fix an optimal solution for I' among those solutions that match $\mathcal{C}'$. Consider some player i and let c'_i denote the configuration that is selected for player i in the optimal solution for I'. We argue that we can find a configuration $c_i \in \mathcal{C}_i$ that

(i) has a total value that is at least a $1/(1+\epsilon)$ fraction of the value of c'_i and

(ii) satisfies $c_i(\bar{v}) \leq c'_i(\bar{v})$ for all $\bar{v} \in \mathcal{T}$.

This gives us a feasible solution for I' that matches $\mathcal{C}$ and has an objective value of at least $\mathrm{OPT}_{\mathcal{C}'}(I')/(1+\epsilon)$ and, thus, proves the statement.

We start by building a configuration c_i independently for every value class $\mathcal{T}_\ell$.

First, we iteratively construct a subset $S_\ell \subseteq \mathcal{T}_\ell$ of value types as follows.

Start with the largest $\bar{v} \in \mathcal{T}_\ell$ such that $c'_i(\bar{v}) > 0$ and add $\bar{v}$ to S_ℓ. Then, find the largest $\bar{v}' \in \mathcal{T}_\ell$ with $\bar{v} > \bar{v}'$ and $c'_i(\bar{v}) < c'_i(\bar{v}')$. Add $\bar{v}'$ to S_ℓ and repeat from $\bar{v}'$ until we do not find another value type to add. Based on S_ℓ, define configuration c_i as $c_i(\bar{v}) = c'_i(\bar{v})$ if $\bar{v} \in S_\ell$ and $c_i(\bar{v}) = 0$ otherwise. By choice of the sets S_ℓ, the configuration c_i is contained in $\mathcal{C}_i$ and satisfies (ii).

It remains to prove that c_i also satisfies (i). Fix an arbitrary value type $\bar{v}_j \in S_\ell$ and let $\bar{v}_{j'}$ denote the next smaller value type in S_ℓ. Recall that the rounded value types are indexed in decreasing order and let s be the integer such that $j' = j + \kappa \cdot (s+1)$. If $\bar{v}_j$ is already the smallest value type in S_ℓ, we set s to the largest integer such that $\ell + \kappa \cdot s \leq \tau$.

We show

$$c_i(\bar{v}_j) \cdot \bar{v}_j \geq \frac{1}{1+\epsilon} \cdot \sum_{s'=0}^{s} c'_i(\bar{v}_{j+s'\cdot\kappa}) \cdot \bar{v}_{j+s'\cdot\kappa}. \tag{B.1}$$

If this inequality holds for all $\bar{v}_j \in S_\ell$, and for all $1 \leq \ell \leq \tau$, then (ii) follows.

To prove the inequality, observe that, by choice of set S_ℓ, all integers $0 \leq s' \leq s$ satisfy $c'_i(\bar{v}_{j+s'\cdot\kappa}) \leq c'_i(\bar{v}_j)$. Furthermore, $\bar{v}_{j+s'\cdot\kappa} = \bar{v}_j/(1+\epsilon)^{\kappa\cdot s'}$ by the rounding of the value types. This gives us

$$\begin{aligned}\sum_{s'=0}^{s} c'_i(\bar{v}_{j+s'\cdot\kappa}) \cdot \bar{v}_{j+s'\cdot\kappa} \leq c'_i(\bar{v}_j) \cdot \sum_{s'=0}^{s} \bar{v}_{j+s'\cdot\kappa} &= c_i(\bar{v}_j) \cdot \bar{v}_j \cdot \sum_{s'=0}^{s} \frac{1}{(1+\epsilon)^{\kappa\cdot s'}} \\ &\leq c_i(\bar{v}_j) \cdot \bar{v}_j \cdot \frac{1}{1 - \frac{1}{(1+\epsilon)^\kappa}} \end{aligned} \tag{B.2}$$

By further using

$$\kappa \geq \frac{1+\epsilon}{\epsilon^2} \geq \frac{1}{\epsilon} \cdot \frac{1}{\ln(1+\epsilon)} \geq \frac{\ln(1+\frac{1}{\epsilon})}{\ln(1+\epsilon)} = \log_{1+\epsilon}\left(\frac{1+\epsilon}{\epsilon}\right),$$

it is not hard to see that (B.2) is at most $c_i(\bar{v}_j) \cdot \bar{v}_j \cdot (1+\epsilon)$, and thus, implies (B.1). This concludes the proof of $\mathrm{OPT}_{\mathcal{C}}(I') \geq \mathrm{OPT}_{\mathcal{C}'}(I')/(1+\epsilon)$. ∎

C Proof of Theorem 1.5

In the job-matroid Makespan problem, the assignment LP can be written as follows, where we have one variable $x_j(i)$ for each pair job j and machine i.

$$\begin{aligned} \sum_{j \in J} x_j(i) \cdot p_{ij} &\leq T \quad && \forall i \in M \\ x_j &\in \mathcal{B}(\mathcal{P}_j) && \forall j \in J \\ x_j(i) &= 0 && \forall j \in J \text{ s.t. } p_{ij} > T \\ x &\geq 0\ , \end{aligned}$$

where $T \geq 0$ is some given makespan bound. We note the subtlety in the assignment LP for the job-matroid Makespan problem (compared to the assignment LP for the resource-matroid SantaClaus problem given in Section 3), where we have an additional constraint for the jobs j with big processing time on some specific machine i. This is sometimes referred to as "parametric pruning" and already appears in the original proof in [25].

Theorem C.1. *Given a fractional assignment x of jobs to machines which is feasible for the assignment LP (with parameter T) of some instance I of the unrelated job-matroid* Makespan *problem, we can obtain, in polynomial time, a feasible integral assignment of jobs to machines of makespan at most $T + \max_{i\in M, j\in J, x_j(i)>0} p_{ij}$. This implies that we can find in polynomial time a feasible integral solution of makespan at most $2\mathrm{OPT}(I)$.*

Proof. The proof here is very similar to the proof of Theorem 3.1. We repeat it here for completeness. For each machine i, we order the jobs in non-increasing order of processing time on machine i, and we denote $\sigma_i(j)$

the index corresponding to the job that appears in j-th position in that order. By definition we have that $p_{i\sigma_i(1)} \geq p_{i\sigma_i(2)} \geq \cdots \geq p_{i\sigma_i(n)}$ for any machine i. W.l.o.g. we also assume that $p_{i\sigma_i(n)} = 0$ for any machine i.

We denote the polymatroids associated to the jobs as $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$, corresponding to the submodular functions $f_1, f_2, \ldots, f_n$. Given the fractional assignment x, we will create a feasible fractional solution x' to a certain polymatroid intersection problem. We define the two polymatroids using a bipartite graph as follows. On the left-hand side, we have a set of vertices W with one vertex w_j for each job j, and on the right-hand side we have a set of vertices U with one vertex u_{ij} for each machine i and the job appearing in jth position in the order σ_i. The edge set will be denoted by E, and both polymatroids will have E as a ground set. We set $E = \{(w_{\sigma_i(j)}, u_{ij})\}_{i\in M, j\in J} \cup \{(w_{\sigma_i(j)}, u_{i(j-1)})\}_{i\in M, j\in J\setminus\{1\}}$ (note the slight change here compared to Theorem 3.1). For some edge $e \in E$, we denote by e_w the job corresponding to its left-hand side vertex, and by e_u the machine corresponding to its right-hand side endpoint. The first polymatroid $\mathcal{P}'_1$ will be associated with the submodular function

$$f_1(S) := \sum_{j=1}^{n} f_j \left(\bigcup_{e\in S: e_w = j} e_u \right) .$$

The second polymatroid $\mathcal{P}'_2$ will be defined using the right-hand side vertices in our graph. Each vertex $u \in U$ will have some degree constraint $d(u)$ and the submodular function f_2 is simply defined as

$$f_2(S) := \sum_{e=(w,u)\in S} d(u) .$$

We define the degree constraints using the following process for each machine i in instance I. We start with

$$d(u_{i1}) = \left\lceil x_{\sigma_i(1)}(i) \right\rceil ,$$

and we define the *remainder* $R_1 := d(u_{i1}) - x_{\sigma_i(1)}(i)$. Then we define recursively the degree constraint $d(u_{ij})$ $(j \geq 2)$ and remainders as follows.

$$\begin{aligned} d(u_{ij}) &:= \left\lceil x_{\sigma_i(j)}(i) - R_{j-1} \right\rceil , \text{ and} \\ R_j &:= \left\lceil x_{\sigma_i(j)}(i) - R_{j-1} \right\rceil - (x_{\sigma_i(j)}(i) - R_{j-1}) = \left\lceil \{x_{\sigma_i(j)}(i)\} - R_{j-1} \right\rceil - (\{x_{\sigma_i(j)}(i)\} - R_{j-1}) . \end{aligned}$$

It is easy to see that the solution x to the assignment LP can be transformed into a feasible fractional solution to our polymatroid intersection problem. Similar to the proof of Theorem 3.1, the remainder R_j is defined to be exactly the quantity by which we can select the edge $(w_{\sigma_i(j)}, u_{i(j-1)})$ in our fractional solution to the polymatroid intersection problem.

Note that this fractional solution to our polymatroid intersection problem is such that we have a basis of the polymatroid $\mathcal{P}'_1$ (the left-hand side polymatroid). By integrality of the polymatroid intersection polytope (see Chapters 46-47 in [27]) there exists an integral solution which is also a basis in $\mathcal{P}'_1$ (and we can find it in polynomial time by finding the maximum cardinality multiset of edges which belongs to the polymatroids intersection).

It is easy to see that this integral solution corresponds to an integral assignment of jobs to machines in the instance I, in which each machine i receives a load of at most

$$\sum_{j=1}^{n} \left\lceil x_{\sigma_i(j)}(i) - R_{j-1} \right\rceil p_{\sigma_i(j)} .$$

Let us compute the difference in objective Δ_i with the fractional solution. We have that

$$\begin{aligned} \Delta_i \leq \sum_{j=1}^{n} (\left\lceil x_{\sigma_i(j)}(i) - R_{j-1} \right\rceil - x_{\sigma_i(j)}(i)) p_{\sigma_i(j)} &= \sum_{j=1}^{n-1} (\left\lceil x_{\sigma_i(j)}(i) - R_{j-1} \right\rceil - x_{\sigma_i(j)}(i)) p_{\sigma_i(j)} \\ &= \sum_{j=1}^{n-1} (\left\lceil \{x_{\sigma_i(j)}(i)\} - R_{j-1} \right\rceil - \{x_{\sigma_i(j)}(i)\}) p_{\sigma_i(j)} , \end{aligned}$$

using $p_{\sigma_i(n)} = 0$. Looking at each term individually, we notice that either $\lceil \{x_{\sigma_i(j)}(i)\} - R_{j-1} \rceil = 0$, or that $\lceil \{x_{\sigma_i(j)}(i)\} - R_{j-1} \rceil = 1$. In the first case the next remainder R_j is equal to $R_{j-1} - \{x_{\sigma_i(j)}(i)\}$. In the second case, we have that

$$R_j = 1 + R_{j-1} - \{x_{\sigma_i(j)}(i)\} \iff \{x_{\sigma_i(j)}(i)\} = 1 + R_{j-1} - R_j \ .$$

Let us denote by J' the set of all indices where the second case happens. Then we can write

$$\begin{aligned}\Delta_i \leq \sum_{j \in J'} p_{\sigma_i(j)} - \sum_{j=1}^{n} \{x_{\sigma_i(j)}(i)\} p_j = \sum_{j \in J'} p_{\sigma_i(j)} - \sum_{j \in J'} p_{\sigma_i(j)} - \sum_{j=1}^{n} (R_{j-1} - R_j) p_{\sigma_i(j)} = \sum_{j=1}^{n} (R_j - R_{j-1}) p_{\sigma_i(j)} \\ = \sum_{j=1}^{n-1} (p_{\sigma_i(j)} - p_{\sigma_i(j+1)}) R_j \leq p_{\sigma_i(1)} \ ,\end{aligned}$$

where we use the fact that $R_0 = p_{\sigma_i(n)} = 0$, and $R_j \leq 1$ for all j. ∎

Theorem 1.5 can be easily obtained by guessing the optimum value makespan T and using Theorem C.1.