On (1,ϵ)-Restricted Max-Min Fair Allocation Problem*†

T.-H. Hubert Chan[1], Zhihao Gavin Tang[2], and Xiaowei Wu[3]

1 **Department of Computer Sicence, The University of Hong Kong, Hong Kong**
hubert@cs.hku.hk
2 **Department of Computer Sicence, The University of Hong Kong, Hong Kong**
zhtang@cs.hku.hk
3 **Department of Computer Sicence, The University of Hong Kong, Hong Kong**
xwwu@cs.hku.hk

Abstract

We study the max-min fair allocation problem in which a set of m indivisible items are to be distributed among n agents such that the minimum utility among all agents is maximized. In the restricted setting, the utility of each item j on agent i is either 0 or some non-negative weight w_j. For this setting, Asadpour et al. [2] showed that a certain configuration-LP can be used to estimate the optimal value within a factor of $4+\delta$, for any $\delta > 0$, which was recently extended by Annamalai et al. [1] to give a polynomial-time 13-approximation algorithm for the problem. For hardness results, Bezáková and Dani [5] showed that it is NP-hard to approximate the problem within any ratio smaller than 2.

In this paper we consider the $(1, \epsilon)$-restricted max-min fair allocation problem, in which for some parameter $\epsilon \in (0, 1)$, each item j is either heavy ($w_j = 1$) or light ($w_j = \epsilon$). We show that the $(1, \epsilon)$-restricted case is also NP-hard to approximate within any ratio smaller than 2. Hence, this simple special case is still algorithmically interesting.

Using the configuration-LP, we are able to estimate the optimal value of the problem within a factor of $3 + \delta$, for any $\delta > 0$. Extending this idea, we also obtain a quasi-polynomial time $(3 + 4\epsilon)$-approximation algorithm and a polynomial time 9-approximation algorithm. Moreover, we show that as ϵ tends to 0, the approximation ratio of our polynomial-time algorithm approaches $3 + 2\sqrt{2} \approx 5.83$.

1998 ACM Subject Classification G.1.2 Approximation, G.1.6 Optimization, G.2.1 Combinatorics

Keywords and phrases Max-Min Fair Allocation, Hypergraph Matching

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.23

1 Introduction

We consider the *Max-Min Fair Allocation problem*. A problem instance is defined by (A, B, w), where A is a set of n agents, B is a set of m items and the utility of each item $j \in B$ perceived by agent $i \in A$ has weight w_{ij}. An allocation of items to agents is $\sigma : B \to A$ such that $\sigma(j) = i$ iff item j is assigned to agent i. The max-min fair allocation problem aims at finding an allocation such that the minimum total weight received by an agent $\min_{i \in A} \sum_{j \in \sigma^{-1}(i)} w_{ij}$ is maximized. The problem is also known as the Santa Claus Problem [4]. In the restricted

* A full version of the paper is available at https://arxiv.org/abs/1611.08060.
† This research is supported in part by the Hong Kong RGC grant 17202715.

version of the problem, it is assumed that each item j has a fixed weight w_j such that for each $i \in A$ and $j \in B$, $w_{ij} \in \{0, w_j\}$, i.e., if an agent has non-zero utility for an item j, the utility is w_j. We focus on this paper the restricted version of the problem (restricted allocation problem) and refer to the problem with general weights the *unrestricted* allocation problem. For the restricted allocation problem, let $B_i = \{j \in B : w_{ij} > 0\}$ be the set of items agent i is interested in. For a collection of items $S \subseteq B$, let $w(S) = \sum_{j \in S} w_j$.

The problem can be naturally formulated as an integer program, with variable x_{ij} for each $i \in A$ and $j \in B$ indicating whether item j is assigned to agent i. Its linear program relaxation *Assignment-LP* (ALP) is shown as below.

$$\begin{aligned} & \max && T \\ \text{s.t.} \quad & \textstyle\sum_{j \in B_i} x_{ij} w_j \geq T, && \forall i \in A \\ & \textstyle\sum_{i \in A} x_{ij} \leq 1, && \forall j \in B \\ & x_{ij} \geq 0, && \forall i \in A, j \in B. \end{aligned}$$

Let OPT be the maximum value of the restricted allocation problem such that in the optimal allocation, every agent is assigned a set of items with total weight at least OPT. Bezáková and Dani [5] showed that any feasible solution x and T for the ALP can be rounded into an allocation such that every agent i receives at least $T - \max_{j \in B_i} w_j$ total value, which implies $\mathsf{OPT} \geq T^* - \max_{j \in B} w_j$, where T^* is the optimal value of the ALP. However, the above result does not yield any guarantee on the integrality gap. Actually, it can be easily shown that the integrality gap of ALP is unbounded since it is possible to have a feasible solution with $T > 0$ while $\mathsf{OPT} = 0$ (e.g., when $|B| < |A|$). It was shown in [5] that it is NP-hard to approximate the problem within any ratio smaller than 2 by a reduction from 3-dimensional matching.

To overcome the limitation of ALP, a stronger linear program called *Configuration-LP* (CLP) was proposed by Bansal and Sviridenko [4], in which an $O(\frac{\log n}{\log \log n})$-approximation algorithm was obtained for the restricted allocation problem. For any $T > 0$, we call an allocation a T-allocation if it assigns to every agent a set of items with total weight at least T.

▶ **Definition 1** (Bundles with Sufficient Utility)**.** For all $i \in A$, the collection of bundles with utility at least T for agent i is $C(i,T) := \{S \subseteq B_i : w(S) \geq T\}$.

The CLP is a feasibility LP associated with T indicating whether it is possible to (fractionally) assign to each agent one unit of bundle with sufficient utility. The LP ($\mathsf{CLP}(T)$) and its dual are shown as follows.

$$\begin{aligned} & \textbf{Primal} && \min && 0 \\ \text{s.t.} \quad & \sum_{S \in C(i,T)} x_{i,S} \geq 1, && && \forall i \in A \\ & \sum_{i, S : j \in S \in C(i,T)} x_{i,S} \leq 1, && && \forall j \in B \\ & x_{i,S} \geq 0, && && \forall i \in A, S \in C(i,T). \end{aligned}$$

$$\begin{aligned} & \textbf{Dual} && \max && \sum_{i \in A} y_i - \sum_{j \in B} z_j \\ \text{s.t.} \quad & y_i \leq \sum_{j \in S} z_j, && && \forall i \in A, S \in C(i,T) \\ & y_i \geq 0, && && \forall i \in A \\ & z_j \geq 0, && && \forall j \in B. \end{aligned}$$

Although $\mathsf{CLP}(T)$ has an exponential number of variables, it is claimed in [4] that the separation problem for the dual LP is the minimum knapsack problem: given a candidate dual solution (y, z), a violated constraint can be identified by finding an agent i and a configuration $S \in C(i,T)$ such that $y_i > \sum_{j \in S} z_j$. Hence, we can solve $\mathsf{CLP}(T)$ to any desired precision. Note that any feasible solution x of $\mathsf{CLP}(T)$ induces a feasible solution $\hat{x}$ for the ALP by setting $\hat{x}_{ij} = \sum_{S : j \in S \in C(i,T)} x_{i,S} \leq 1$ for all $i \in A$ and $j \in B$.

▶ **Definition 2** (Integrality Gap). Let T^* be the maximum value such that $\mathsf{CLP}(T^*)$ is feasible. The ratio $\frac{T^*}{\mathsf{OPT}}$ is known as the *integrality gap*.

Note that any upper bound c for the integrality gap implies that we can estimate the optimal value of the problem within a factor of $c + \delta$, for any $\delta > 0$. It is shown in [4] that the integrality gap of CLP for the unrestricted allocation problem is bounded by $O(\sqrt{n})$. By repeatedly using the Lovasz Local Lemma, Uriel Feige [8] proved that the integrality gap of CLP for the restricted allocation problem is bounded by a constant. The result was later turned into a constructive proof by Haeupler [11], who obtained the first constant approximation algorithm for the restricted allocation problem, although the constant is unspecified. The integrality gap of CLP was later shown in [2] to be no larger than 4 by a local search technique developed from Haxell [12] for finding perfect matchings in bipartite hypergraphs. However, the algorithm is not guaranteed to terminate in polynomial time. It is later shown by Polacek and Svensson [15] that a simple modification of the local search algorithm can improve the running time from $2^{O(n)}$ to $n^{O(\log n)}$, which implies a quasi-polynomial $(4 + \delta)$-approximation algorithm, for any $\delta > 0$. Very recently, Annamalai et al. [1] further extended the local search technique developed in [2, 15] for the restricted allocation problem and obtained a polynomial-time 13-approximation algorithm for the problem.

1.1 The $(1, \epsilon)$-Restricted Allocation Problem

We consider in this paper the $(1, \epsilon)$-restricted allocation problem, in which for some $\epsilon \in (0, 1)$, each item $j \in B$ is either *heavy* ($w_j = 1$) or *light* ($w_j = \epsilon$). As the simplest case of the allocation problem, the problem is not well understood. The current best approximation results for the problem are for the restricted allocation problem. Indeed, we believe that a better understanding of the $(1, \epsilon)$-restricted setting will shed light on improving the restricted (and even the unrestricted) allocation problem.

The $(1, \epsilon)$-restricted setting has been studied under different names. Golovin [10] studied the "Big Goods/Small Goods" max-min allocation problem, which is exactly the same as the problem we consider in this paper, in which a small item has weight either 0 or 1 for each agent; a big item has weight either 0 or $x > 1$ for each agent. They gave an $O(\sqrt{n})$-approximation algorithm for this problem and proved that it is NP-hard to approximate the "Big Goods/Small Goods" max-min allocation problem within any ratio smaller than 2 by giving a hard instance with $x = 2$. We show in this paper that the inapproximability result holds for any fixed $x \geq 2$ by generalizing the hardness instance shown in [5]. Later Khot and Ponnuswami [13] generalized the "Big Goods/Small Goods" setting and considered the $(0, 1, U)$-max-min allocation problem with sub-additive utility function in which the weight of an item to an agent is either 0, 1 or U for some $U > 1$ and obtained an $\frac{n}{\alpha}$-approximation algorithm with $m^{O(1)}n^{O(\alpha)}$ running time, for any $\alpha \leq \frac{n}{2}$. Note that in their setting an item can have weight 1 for an agent and U for another. In the seminal paper, Bansal and Sviridenko [4] obtained an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the restricted allocation problem by first reducing the problem to the $(1, \epsilon)$-restricted case for an arbitrarily small $\epsilon > 0$ while losing a constant factor on the approximation ratio, and then proving an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the $(1, \epsilon)$-restricted case.

The max-min fair allocation problem is closely related to the problem of scheduling jobs on *unrelated machines* to minimize *makespan*, which we call the *min-max allocation problem*. The problem has the same input as the max-min fair allocation problem but aims at finding an allocation that minimizes $\max_{i \in A} \sum_{j \in \sigma^{-1}(i)} w_{ij}$. Lenstra et al. [14] showed a

2-approximation algorithm for the min-max allocation problem by rounding the ALP for the problem. Applying the techniques developed for the max-min fair allocation problem, Svensson [16] gave a $\frac{5}{3}+\epsilon$ upper bound for the CLP's integrality gap of the $(1,\epsilon)$-restricted min-max allocation problem and then extended it to a 1.9412 upper bound for the general case. However, their algorithm is not known to converge in polynomial time. Recently Chakrabarty et al. [7] obtained the first $(2-\delta)$-approximation algorithm for the $(1,\epsilon)$-restricted min-max allocation problem, for some constant $\delta > 0$. They considered the case when ϵ is close to 0 since it is easy to obtain a $(2-\epsilon)$-approximation algorithm for the $(1,\epsilon)$-restricted min-max allocation problem.

Since the $(1,\epsilon)$-restriction is considered in the community to be interesting for the min-max setting, in this paper we consider this restriction for the max-min setting.

1.2 Summary of Our Results

We first show that we can slightly improve the hardness result of Golovin [10] for the $(1,\epsilon)$-restricted allocation problem. Note that in the unweighted case ($\epsilon = 1$), the problem can be solved in polynomial time by combining the max-flow computation between A and B, with a binary search on the optimal value. The above algorithm for the unweighted case actually provides a trivial $\frac{1}{\epsilon}$-approximation algorithm for the $(1,\epsilon)$-restricted allocation problem. Hence, we have a polynomial-time algorithm with ratio $\frac{1}{\epsilon} < 2$ for the problem when $\epsilon > 0.5$.

▶ **Theorem 3** (Inapproximability). *For any $\epsilon \leq 0.5$, it is* NP*-hard to approximate the $(1,\epsilon)$-restricted allocation problem within any ratio smaller than* 2*.*

The proof is included in our full version. Our reduction shows that it is NP-hard to estimate the optimal value of the problem within any ratio smaller than 2. The above hardness result implies that the integrality gap of $\mathsf{CLP}(T)$ is at least 2 unless $\mathsf{P} = \mathsf{NP}$. However, we can remove the $\mathsf{P} \neq \mathsf{NP}$ assumption by giving a hard instance explicitly (in the full version).

For the restricted allocation problem, the best hardness result on the approximation ratio is 2 while the best upper bound for the integrality gap of $\mathsf{CLP}(T)$ is 4. It is not known which bound (or none) is tight. As a step towards closing this gap, we analyze the integrality gap of $\mathsf{CLP}(T)$ for the $(1,\epsilon)$-restricted case and show that the upper bound of 4 is not tight in this case (Section 2). Our upper bound implies that in polynomial time we can estimate OPT for the $(1,\epsilon)$-restricted allocation problem within a factor of $3+\delta$, for any $\delta > 0$.

▶ **Theorem 4** (Integrality Gap). *The integrality gap of the configuration-LP of the $(1,\epsilon)$-restricted allocation problem is at most* 3*.*

We also observe that by picking the "closest addable edge", the running time of the local search algorithm can be improved to quasi-polynomial (Section 3). The idea was first used by Polacek and Svensson [15] to obtain the $(4+\delta)$-approximation algorithm for the restricted allocation problem. However, instead of constructing feasible dual solutions for $\mathsf{CLP}(T)$, our analysis is based on the assumption of $T \leq \mathsf{OPT}$ and is a direct extension of our proof on the integrality gap of $\mathsf{CLP}(T)$.

▶ **Theorem 5** (Quasi-Polynomial-Time Approximation). *There exists a $(3+4\epsilon)$-approximation algorithm for the $(1,\epsilon)$-restricted allocation problem that runs in $n^{O(\frac{1}{\epsilon}\log n)}$ time.*

We further extend the quasi-polynomial approximation algorithm by combining the lazy update idea of [1] to obtain a polynomial approximation algorithm (Section 4).

▶ **Theorem 6** (Polynomial-Time Approximation). *For any $\epsilon \in (0,1)$, there exists a polynomial-time 9-approximation algorithm for the $(1,\epsilon)$-restricted allocation problem. Moreover, the approximation ratio approaches $3+2\sqrt{2} \approx 5.83$ as ϵ tends to 0.*

Interestingly, while our quasi-polynomial- and polynomial-time algorithms are extended from the integrality gap analysis by combining ideas on improving the running time of local search, unlike existing techniques, our algorithms and analysis do not directly use the feasibility of $\mathsf{CLP}(T)$. To lead to contradictions, existing results [15, 1] tried to construct feasible dual solutions for $\mathsf{CLP}(T)$ with positive objective values (which implies the infeasibility of $\mathsf{CLP}(T)$). In contrast, our analysis shows that as long as $T \le \mathsf{OPT}$, our algorithms terminate with the claimed approximation ratios, which simplifies the analysis and is an advantage in some cases when $\mathsf{CLP}(T)$ cannot be applied, e.g., when the utility function is sub-additive [13].

1.3 Other Related Work

Unrestricted Allocation Problem. Based on Bansal and Sviridenko's proof [4] of $O(\sqrt{n})$-integrality gap for the unrestricted allocation problem, Asadpour and Saberi [3] achieved an $\tilde{O}(\sqrt{n})$-approximation algorithm. The current best approximation result for the problem is an $\tilde{O}(n^{\delta})$-approximation algorithm that runs in time $n^{O(\frac{1}{\delta})}$, for any $\delta = \Omega(\frac{\log\log n}{\log n})$, obtained by Chakrabarty et al. [6].

Other Utility Functions. The max-min fair allocation problem with different utility functions has also been considered. Golovin [10] gave an $(m-n+1)$-approximation algorithm for the problem when the utility functions of agents are submodular. We note that their result can also be extended to sub-additive utility functions. Khot and Ponnuswami [13] also considered the problem with sub-additive utility functions and obtained a $(2n-1)$-approximation algorithm. Later Goemans and Harvey [9] obtained an $\tilde{O}(n^{\frac{1}{2}+\delta})$-approximation for submodular max-min allocation problem in $n^{O(\frac{1}{\delta})}$ time using the $\tilde{O}(n^{\delta})$-approximation algorithm by Chakrabarty et al. [6] as a black box.

2 Integrality Gap for Configuration LP

We show in this section that for the $(1,\epsilon)$-restricted allocation problem, the integrality gap of the CLP is at most 3. Fix $T > 0$ be such that $\mathsf{CLP}(T)$ is feasible.

We show that whenever $\mathsf{CLP}(T)$ is feasible, there exists a $\frac{T}{3}$-allocation (hence $\mathsf{OPT} \ge \frac{T}{3}$), which implies an integrality gap of at most 3. Given any solution x for $\mathsf{CLP}(T)$ and the induced ALP solution $\hat{x}$, for all $\hat{x}_{ij} = 0$, we can remove j from B_i (pretending that i is not interested in j). This operation will preserve the feasibility of x while (possibly) decreasing OPT, which could only enlarge the integrality gap. From now on we assume that a positive fraction of every item in B_i is assigned to agent i.

Assumption on T: To achieve a $\frac{T}{3}$-allocation, we can assume that $T < \frac{3}{2}$; otherwise, we can get a $T-1 \ge \frac{T}{3}$ allocation by rounding the ALP solution $\hat{x}$ [5]. We can further assume $T \ge 1$ since otherwise we can set all weights $w_j \ge T$ to T (which does not change $\mathsf{CLP}(T)$) and scale all weights so that the maximum weight is 1. From now on, we assume that $T \in [1, \frac{3}{2})$ and $\mathsf{CLP}(T)$ is feasible.

Let $k = \lceil \frac{T}{\epsilon} \rceil$. Note that every bundle consisting solely of light items must contain at least k items to have sufficient utility. For all $i \in A$, let $B_i^1 = \{j \in B_i : w_j = 1\}$ be the set

of heavy items and $B_i^\epsilon = \{j \in B_i : w_j = \epsilon\}$ be the set of light items. Our algorithm fixes an integer $r = \lceil \frac{k}{3} \rceil$ and tries to assign items such that each agent i receives either a heavy item $j \in B_i^1$ or r light items in B_i^ϵ. Suppose we are able to find such an allocation, then the integrality gap is $\frac{T}{r\epsilon} \leq 3$.

2.1 Getting a "Minimal" Solution

Let x^* be a solution for $\mathsf{CLP}(T)$. We create another solution x (which might not be feasible) as follows. Initialize $x_{i,S} = 0$ for all $i \in A$ and $S \subseteq B_i$. For all $x^*_{i,S} > 0$, where $S \in C(i,T)$,

1. if $S' = S \cap B_i^1 \neq \emptyset$, then set $x_{i,S'} = x^*_{i,S}$;
2. otherwise, S contains only light items and set $x_{i,S} = x^*_{i,S}$.

Note that for each $i \in A$ we have the following properties on x:

1. (heavy/light configurations) if $x_{i,S} > 0$, then $(S \subseteq B_i^1 \wedge |S| \geq 1)$ or $(S \subseteq B_i^\epsilon \wedge |S| \geq k)$.
2. (covering constraint for agent) $\sum_{S \subseteq B_i} x_{i,S} = \sum_{S \in C(i,T)} x^*_{i,S} \geq 1$.
3. (packing constraint for item) for all $j \in B$: $\sum_{i,S:j \in S} x_{i,S} \leq \sum_{i,S:j \in S \in C(i,T)} x^*_{i,S} \leq 1$.

Now we construct a hypergraph $H(A \cup B, E)$ as follows: for all $x_{i,S} > 0$,

1. if $S \subseteq B_i^1$, then for each $j \in S$, add $\{i,j\}$ to E (we call such an edge **heavy**);
2. otherwise for each $S' \subseteq S$ and $|S'| = r$, add $\{i\} \cup S'$ to E (we call such an edge **light**).

A matching $M \subseteq E$ is a collection of disjoint edges. Note that any perfect matching of H that matches all nodes in A provides an allocation that assigns each $i \in A$ either a heavy item or r light items. For all $F \subseteq E$, let $A(F) = A \cap (\bigcup_{e \in F} e)$ and $B(F) = B \cap (\bigcup_{e \in F} e)$.

2.2 Finding a Perfect Matching

Recall that the existence of a perfect matching in $H(A \cup B, E)$ such that every agent in A is matched implies that the integrality gap of $\mathsf{CLP}(T)$ is at most 3.

▶ **Theorem 7.** *The above hypergraph $H(A \cup B, E)$ has a perfect matching.*

Proof. Given a partial matching $M \subseteq E$, we show how to extend its cardinality by one if $|M| \leq |A| - 1$. Let $i_0 \in A \backslash A(M)$ be an agent not matched by M. For the initial step, suppose X_1 contains an arbitrary edge e_1 with $A(e_1) = \{i_0\}$ and $Y_1 = \mathsf{blocking}(e_1) = \{f \in M : B \cap e_1 \cap f \neq \emptyset\}$ be the blocking edges of e_1. If $\mathsf{blocking}(e_1) = \emptyset$, then we can add edge e_1 to the matching. Assume $\mathsf{blocking}(e_1) \neq \emptyset$.

For the recursive step, suppose we already have edges X_t (where $t = |X_t|$) and Y_t, which together form an alternating tree rooted at i_0. We consider adding the $(t+1)$-st edge to X_t as follows. An edge $e \in E$ is **addable** if (1) $A(e) \in A(X_t \cup Y_t)$; (2) $B(e) \cap B(X_t \cup Y_t) = \emptyset$. If such an edge e_{t+1} exists and $\mathsf{blocking}(e_{t+1}) \neq \emptyset$, let $X_{t+1} = X_t \cup \{e_{t+1}\}$ and $Y_{t+1} = Y_t \cup \mathsf{blocking}(e_{t+1})$. If $\mathsf{blocking}(e_{t+1}) = \emptyset$, then we **contract** X_t by swapping out blocking edges (the details of contraction will be discussed later). The contraction operation guarantees that every addable edge has at least one blocking edge.

▶ **Claim 8** (Always Addable)**.** *There is always an addable edge e_{t+1}.*

Proof. Let $P = A(X_t \cup Y_t)$ be the agents in the tree. Note that $|P| = |Y_t| + 1$ since each agent $i \neq i_0$ in P has an unique blocking edge that **introduces** i.

Let X_t^1 (Y_t^1) be the heavy edges and X_t^ϵ (Y_t^ϵ) be the light edges of X_t (Y_t).

We have $|X_t^1| = |Y_t^1|$ since heavy edges can only be blocked by heavy edges. We have $|X_t^\epsilon| \leq |Y_t^\epsilon|$ since each addable edge has at least one blocking edge.

Let $x_P^1 = \sum_{i \in P} \sum_{S \subseteq B_i^1} x_{i,S}$ be the total units of heavy bundles assigned to P by x, which is a lower bound for the total number of heavy items $B_P^1 = \cup_{i \in P} B_i^1$ agents in P are interested in since

$$x_P^1 = \sum_{i \in P} \sum_{S \subseteq B_i^1} x_{i,S} \leq \sum_{i \in P} \sum_{S \subseteq B_i^1} \sum_{j \in S} x_{i,S} = \sum_{j \in B_P^1} \sum_{i,S: j \in S \subseteq B_i^1} x_{i,S} \leq |B_P^1|.$$

Let $x_P^\epsilon = \sum_{i \in P} \sum_{S \subseteq B_i^\epsilon} x_{i,S}$ be the total units of light bundles assigned to P by x. By construction of x, we have

$$\sum_{i \in P} \sum_{S \subseteq B_i} x_{i,S} = \sum_{i \in P} \sum_{S \subseteq B_i^1} x_{i,S} + \sum_{i \in P} \sum_{S \subseteq B_i^\epsilon} x_{i,S} = x_P^1 + x_P^\epsilon \geq |P|.$$

Since $|Y_t^1|$ heavy items are already introduced in the tree, if $x_P^1 > |Y_t^1|$, then there must exist an addable heavy edge for some $i \in P$. If $x_P^1 \leq |Y_t^1|$, then we have $x_P^\epsilon \geq |P| - x_P^1 \geq |Y_t^\epsilon| + 1 \geq |X_t^\epsilon| + 1$. Note that every light addable edge has at most $r-1$ unblocked items, the total number of light items in the tree is

$$|B^\epsilon(X_t \cup Y_t)| \leq (r-1)|X_t^\epsilon| + r|Y_t^\epsilon| \leq (2r-1)(x_P^\epsilon - 1) < (2r-1)x_P^\epsilon. \tag{1}$$

For each $i \in P$ and $S \subseteq B_i^\epsilon$, if $x_{i,S} > 0$, then by construction we have $|S| \geq k \geq 3r-2$. If i has no more addable light edges (has at most $r-1$ **unintroduced** light items in H), then at least $\sum_{S \subseteq B_i^\epsilon} (|S| - (r-1)) x_{i,S} \geq (2r-1) \sum_{S \subseteq B_i^\epsilon} x_{i,S}$ units of configurations of light items appear in the tree. If there is no more addable light edges for all $i \in P$, then we have

$$|B^\epsilon(X_t \cup Y_t)| \geq \sum_{j \in B^\epsilon(X_t \cup Y_t)} \sum_{i,S: j \in S \subseteq B_i^\epsilon} x_{i,S} \geq (2r-1) \sum_{i \in P} \sum_{S \subseteq B_i^\epsilon} x_{i,S} = (2r-1) x_P^\epsilon,$$

which is a contradiction to (1). ◀

Contraction: If $\mathsf{blocking}(e_{t+1}) = \emptyset$, then we remove the blocking edge f that introduces $A(e_{t+1})$ from the matching and include e_{t+1} into the matching. Both e_{t+1} and f are removed from the tree. We also remove all edges added after f since they can possibly be introduced by $A(f)$. We call this operation a contraction on e_{t+1}. Note that this operation reduces the size of $\mathsf{blocking}(e')$ by one, for the edge e' that is blocked by f. If $\mathsf{blocking}(e') = \emptyset$ after that, then we further contract e' recursively. After all contractions, suppose the remaining addable edges in the tree are $e_1, e_2, \ldots, e_{t'}$ (ordered by the time they are added to the tree), we set $t = t'$, $X_{t'}$ and $Y_{t'}$ be the addable and blocking edges, respectively.

Signature: At any moment before including an addable edge (suppose there are t addable edges in the tree), let $s_i = |\mathsf{blocking}(e_i)|$ for all $i \in [t]$. Let $\mathbf{s} = (s_1, s_2, \ldots, s_t, \infty)$ be the signature of the tree. Then, we have the following.

1. The lexicographical value of $\mathbf{s}$ reduces after each iteration. If there is no contraction in the iteration, then in the signature, the $(t+1)$-st coordinate decreases from ∞ to s_{t+1}, while s_i remains the same for all $i \leq t$. Otherwise, let e_i be the edge whose number of blocking edges is reduced by one but remains positive in the contraction phase. Then, we have s_i is reduced by one while s_j remains the same for all $j < i$.
2. There are at most 2^n different signatures since $\sum_{i \in [t]} s_i \leq n$ and $t \leq n$.

Since an addable edge can be found in polynomial time and the contraction operation stops in polynomial time, a perfect matching can be found in $n \cdot 2^n \cdot \mathsf{poly}(n)$ time. ◀

3 Quasi-Polynomial-Time Approximation Algorithm

We show in this section that a simple modification on the algorithm for finding a perfect matching in Section 2 can dramatically improve the running time from $2^{O(n)}$ to $n^{O(\log n)}$. Assume that $T \leq \mathsf{OPT}$. Note that in this case we can still assume $T \in [1, \frac{3}{2})$.

Note that combining the polynomial time $\frac{1}{\epsilon}$-approximation algorithm, the approximation ratio we obtain in quasi-polynomial time is $\min\{\frac{1}{\epsilon}, 3+4\epsilon\} \leq 4$ for all $\epsilon \in (0,1)$. Moreover, when $\epsilon \to 0$ (in which case the problem is still $(2-\delta)$-inapproximable), our approximation ratio approaches the integrality gap upper bound 3.

Proof of Theorem 5. Let T be a guess of OPT and $k = \lceil \frac{T}{\epsilon} \rceil$. Since the statement trivially holds for $\epsilon \geq \frac{1}{4}$ ($\frac{1}{\epsilon} \leq 3+4\epsilon$). We assume that $\epsilon < \frac{1}{4}$ (hence $k \geq 5$). We show that if $T \leq \mathsf{OPT}$, then we can find in quasi-polynomial time a $\frac{T}{3+4\epsilon}$-allocation; if no such allocation is found after the time limit, then T should be decreased as in binary search. Let $r = \lceil \frac{k}{3+4\epsilon} \rceil$. It suffices to show that a feasible allocation that assigns to each agent i either a heavy item in B_i^1 or r light items in B_i^ϵ can be found in $n^{O(\log n)}$ time, for any $\epsilon < \frac{1}{4}$. We define a heavy edge $\{i,j\}$ for each $j \in B_i^1$ and a light edge $\{i\} \cup S$ for each $S \subseteq B_i^\epsilon$ and $|S| = r$.

As in the proof of Theorem 7, we wish to find a perfect matching for all agents in A. Suppose in some partial matching, there is an unmatched agent i_0 and we construct an alternating tree rooted at i_0. For each addable edge e, we denote by $d(e)$ the number of light edges (including e) in the **path** from i_0 to e in an alternating tree rooted at i_0. Note that a path is a sequence of edges alternating between addable edges and blocking edges. The algorithm we use in this section is the same as previous, except that when there are addable edges, we always pick the one e such that the **distance** $d(e)$ is **minimized**. We show that in this case there is always an addable edge within distance $O(\frac{1}{\epsilon}\log n)$.

Let X_i and Y_i be the set of addable edges and blocking edges at distance i from i_0, respectively. Note that $Y_i = \emptyset$ for all odd i since light blocking edge must be introduced due to light addable edge. Moreover, since on the path from i_0 to every addable edge $e \in X_i$, the light edge (if any) closest to e must be a blocking edge (of even distance), we know that X_{odd} contains only light edges and X_{even} contains only heavy edges. Let Y_i^1 and Y_i^ϵ be the set of heavy edges and light edges in Y_i, respectively.

Let $L = \lceil \log_{1+\frac{\epsilon}{10}} n \rceil$. It suffices to prove Claim 9 below since it implies that

$$|Y^\epsilon_{\leq 2L+2}| > (1+\frac{\epsilon}{10})|Y^\epsilon_{\leq 2L}| > (1+\frac{\epsilon}{10})^L |Y_2^\epsilon| \geq n,$$

which is a contradiction and implies that there is always an addable edge within distance $2L+1$. Note the the last inequality also comes from Claim 9 since otherwise $|Y_2^\epsilon| = 0$ and $|Y_4^\epsilon| = 0 \leq \frac{\epsilon}{10}|Y_2^\epsilon|$ would be a contradiction.

▶ **Claim 9.** *For all $l \in [L]$, when there is no more addable edge within distance $2l+1$, we have $|Y^\epsilon_{2l+2}| > \frac{\epsilon}{10}|Y^\epsilon_{\leq 2l}|$.*

Proof. Let $P = A(X_{\leq 2l} \cup Y_{\leq 2l}) = A(Y_{\leq 2l}) \cup \{i_0\}$. Since there is no more addable edges within distance $2l+1$, we know that every agent $i \in P$ does not admit any addable edges. Hence for each $i \in P$, all heavy items in B_i^1 are already included in $B^1(X^1_{\leq 2l})$ and at most $r-1$ light items in B_i^ϵ are not included in $B^\epsilon(X^\epsilon_{\leq 2l+1} \cup Y^\epsilon_{\leq 2l+2})$.

Since $T \leq \mathsf{OPT}$, we know that at least $|P| - |B^1(X^1_{\leq 2l})| = |Y^\epsilon_{\leq 2l}| + 1$ agents in P are assigned only light items. Hence, out of at least k light items assigned to each of those agents, at least $k-r+1$ items must be included in $B^\epsilon(X^\epsilon_{\leq 2l+1} \cup Y^\epsilon_{\leq 2l+2})$, which means

$$|B^\epsilon(X^\epsilon_{\leq 2l+1} \cup Y^\epsilon_{\leq 2l+2})| \geq (k-r+1)(|Y^\epsilon_{\leq 2l}| + 1).$$

Assume $|Y^\epsilon_{2l+2}| \le \frac{\epsilon}{10}|Y^\epsilon_{\le 2l}|$, we have $|Y^\epsilon_{\le 2l+2}| \le (1+\frac{\epsilon}{10})|Y^\epsilon_{\le 2l}|$. Since every addable edge contains at most $r-1$ unblocked items (items not used by M), we have the following upper bound for the number of light items in the tree:

$$|B^\epsilon(X^\epsilon_{\le 2l+1} \cup Y^\epsilon_{\le 2l+2})| \le (r-1)|X^\epsilon_{\le 2l+1}| + r|Y^\epsilon_{\le 2l+2}| \le (1+\frac{\epsilon}{10})(2r-1)|Y^\epsilon_{\le 2l}|.$$

For $\epsilon < \frac{1}{4}$, $T \in [1, \frac{3}{2})$, $k = \lceil \frac{T}{\epsilon} \rceil$ and $r = \lceil \frac{k}{3+4\epsilon} \rceil$, we have $k \ge 3\lceil \frac{k}{3} \rceil - 2 \ge 3r-2$. Suppose $k = 3r-2$, then we have

$$k = 3\lceil \frac{k}{3+4\epsilon} \rceil - 2 \le \frac{3k}{3+4\epsilon} + 1 = k - (\frac{4\epsilon k}{3+4\epsilon} - 1),$$

which is a contradiction since $\frac{4\epsilon k}{3+4\epsilon} > 1$. Hence we have $k \ge 3r-1$, which implies $k - r + 1 \ge (3r-1) - (r-1) = 2r \ge (1+\frac{\epsilon}{10})(2r-1)$ since $r \le \frac{5}{\epsilon}$. Hence we have a contradiction. ◀

At any moment before adding an addable edge, suppose we have constructed $X_{\le 2l}$ and $Y_{\le 2l}$. By the above argument we have $2l \le 2L = O(\frac{1}{\epsilon}\log n)$. Let $a_i = -|X_i|$. Let $|Y^1_i| = b_i$ and $|Y^\epsilon_i| = b_{i-1}$ for all even i. Let $\mathbf{s} = (a_0, b_0, a_1, b_1, \ldots, a_{2l}, b_{2l}, \infty)$ be the **signature** of the alternating tree. We show that $\mathbf{s}$ is lexicographically decreasing accross all iterations.

No contraction: Suppose we added an addable edge e with $\mathsf{blocking}(e) \ne \emptyset$, then e will be included in $X_{\le 2l}$ or a newly constructed X_{2l+1}, in both cases the lexicographic value of $\mathbf{s}$ decreases since the last modified coordinate decreases.

Contraction: Suppose the newly added edge has no blocking edge, then in the contraction, let $f \in Y^\epsilon_{2i}$, which must be light, be the last blocking edge that is removed. Since b_{2i-1} decreases while a_j (for all $j \le 2i-1$) and b_j (for all $j \le 2i-2$) do not change, the lexicographic value of $\mathbf{s}$ decreases.

Since $L = O(\frac{1}{\epsilon}\log n)$, there are $n^{O(\frac{1}{\epsilon}\log n)}$ different signatures. Since an addable edge can be found in polynomial time and the contraction operation stops in polynomial time, the running time of the algorithm is $n \cdot \mathsf{poly}(n) \cdot n^{O(\frac{1}{\epsilon}\log n)} = n^{O(\frac{1}{\epsilon}\log n)}$. ◀

4 Polynomial-Time Approximation Algorithm

We give a polynomial-time approximation algorithm in this section. Based on the previous analysis, to improve the running time from $n^{O(\log n)}$ to $n^{O(1)}$, we need to bound the total number of iterations (signatures) by $\mathsf{poly}(n)$. On a high level, our algorithm is similar to that of Annamalai et al. [1]: we apply the idea of lazy update and greedy player such that after each iteration, either a new layer is constructed or the size of the highest layer changed is reduced by a constant factor. However, instead of constructing feasible dual solutions, we extend the charging argument used in the previous sections on counting the number of light items in the tree to prove the exponential growth property of the alternating tree.

In binary search, let T be a guess of OPT. As explained earlier, we can assume $T \in [1, \frac{3}{2})$. Let $k = \lceil \frac{T}{\epsilon} \rceil$. Our algorithm aims at assigning to each agent either a heavy item or r light items, for some fixed $r \le \frac{k}{2}$ when $T \le \mathsf{OPT}$. Such an allocation gives a $\frac{k}{r}$-approximation. Let $p \in (r, k)$ be an integer parameter. Let $0 < \mu \ll 1$ be a very small constant, e.g., $\mu = 10^{-10}$.

As before, for each $i \in A$, we call $\{i, j\}$ a heavy edge for $j \in B^1_i$, and $\{i\} \cup S$ a light edge if $S \subseteq B^\epsilon_i$. However, in this section, we use two types of light edges: either $|S| = p$ (addable edges) or $|S| = r$ (blocking edges). Let M be a maximum matching between A and B^1. We can regard M as a partial allocation that assigns maximum number of heavy items. Let i_0

be an unmatched node in M. We can further assume that every heavy item is interesting to at least 2 agents since otherwise we can assign it to the only agent and remove the item and the agent from the problem instance. We use " $+$ " and " $-$ " to denote the inclusion and exclusion of singletons in a set, respectively.

4.1 Flow Network

Let $G(A \cup B^1, E_M)$ be a **directed** graph uniquely defined by M as follows. For all $i \in A$ and $j \in B_i^1$, if $\{i,j\} \in M$ then $(j,i) \in E_M$, otherwise $(i,j) \in E_M$. We can interpret the digraph as the residual graph of the "interest" network (a digraph with directed edges from each i to $j \in B_i^1$) with current flow M. The digraph G has the following properties

- every $i \in A$ has in-degree ≤ 1, every $j \in B^1$ has out-degree ≤ 1 and in-degree ≥ 1.
- all heavy items reachable from $i \in A$ with in-degree 0 must have out-degree 1 (otherwise we can augment the size of M by one).

Given two sets of light edges Y and X ($A(Y)$ and $A(X)$ do not have to be disjoint), let $f(Y,X)$ denote the maximum number of node-disjoint paths in $G(A \cup B^1, E_M)$ from $A(Y)$ to $A(X)$. Let $F(Y,X)$ be those paths. We will later see that each such path alternates between heavy edges and their blocking edges. Unlike the quasi-polynomial-time algorithm, in our polynomial-time algorithm, the heavy edges do not appear in the alternating tree. Instead, they are used in the flow network $G(A \cup B^1, E_M)$ to play a role of connecting existing addable light edges and blocking light edges.

4.2 Building Phase

▶ **Definition 10** (Layers). For all $i \geq 1$, a layer L_i is a tuple (X_i, Y_i), where X_i is a set of addable edges and Y_i is a set of blocking edges that block edges in X_i.

Initialize $l = 0$, $L_0 = (\emptyset, \{(i_0, \emptyset)\})$. We call an addable edge $e = \{i\} \cup P$ **unblocked** if it contains at least r unblocked light items: $|P \backslash (\bigcup_{e' \in \mathsf{blocking}(e)} B^\epsilon(e'))| \geq r$. Initialize the set of unblocked addable edges be $I = \emptyset$. Throughout the whole algorithm, we maintain a set I of unblocked addable edges and layers $L_i(X_i, Y_i)$ for all $i \leq l$, where X_i contains blocked addable edges. Initialize $X_{l+1} = Y_{l+1} = \emptyset$. We build a new layer as follows.

▶ **Definition 11** (Addable). Given layers $X_{\leq l+1}$ and $Y_{\leq l}$, an edge $e = \{i\} \cup P$ is addable if $|P| = p$ and $P \subseteq B_i^\epsilon \backslash B^\epsilon(X_{\leq l+1} \cup Y_{\leq l})$ such that $f(Y_{\leq l}, X_{\leq l+1} \cup I + e) > f(Y_{\leq l}, X_{\leq l+1} \cup I)$.

Note that such an edge is connected to a blocking edge in $Y_{\leq l}$ by a path in $G(A \cup B^1, E_M)$ that is disjoint from other paths connecting existing blocking edges and addable edges.

Given an addable edge: if it is unblocked, then include it in I; otherwise include it in X_{l+1}. When there is no more addable edges, let $Y_{l+1} = \mathsf{blocking}(X_{l+1}) = \bigcup_{e \in X_{l+1}} \mathsf{blocking}(e)$, set $l = l + 1$ and try to contract L_l. Note that a blocking edge can block multiple addable edges.

4.3 Collapse Phase

Let $W = F(Y_{\leq l}, I)$ be constructed as follows. Initialize $W = \emptyset = F(Y_{\leq 0}, I)$. Recursively for $i = 1, 2, \ldots, l$, let $W = F(Y_{\leq i}, I)$ be augmented from $W = F(Y_{\leq i-1}, I)$. In the final W, let $W_i \subseteq W$ be the paths from $A(Y_i)$ to $A(I)$ and let $I_i \subseteq I$ be those reached by W_i. By the above construction, if $f \in Y_{\leq i}$ have no out-flow in $F(Y_{\leq i}, I)$, then it will not have out-flow in $F(Y_{\leq j}, I)$, for any $j > i$. Hence we have for all $i = 1, 2, \ldots, l$, $|W_i| = |I_i|$ and $|W_{\leq i}| = |I_{\leq i}| = f(Y_{\leq i}, I) = f(Y_{\leq i}, I_{\leq i})$. Note that every path in W_i starts with an agent

$u \in A(Y_i)$ that is assigned a light edge by M and ends at a agent $v \in A(I_i)$ with an unblocked addable edge, which provides a possibility of swapping out a blocking edge in the tree with an unblocked addable edge (by reassigning all heavy items in the path).

▶ **Definition 12** (Collapsible). We call layer L_i collapsible if $|I_i| \geq \mu |Y_i|$.

Intuitively, $|I_i| \geq \mu |Y_i|$ implies that we can swap out a μ fraction of blocking edges in Y_i (which is called a collapse). Let L_t be the earliest collapsible layer, we collapse it as follows.

Step-(1). For each path $P(u, v)$ in W_t from $e_1 := \{u\} \cup R \in Y_t$ to $e_2 := \{v\} \cup P \in I_t$:

1. $M = M - e_1 + e'$, swap out blocking edge e_1 with $e' := \{v\} \cup P'$, where $|P'| = r$ and $P' \subseteq P \setminus \bigcup_{e \in \mathsf{blocking}(e_2)} B^{\epsilon}(e)$,
2. reverse all heavy edges in $P(u, v)$: $M = M \cup \{\{i, j\} : (i, j) \in P(u, v) \cap (A \times B)\} \setminus \{\{i', j'\} : (j', i') \in P(u, v) \cap (B \times A)\}$.

Note that after the above operations, only Y_t and M are changed: the size $|Y_t|$ is reduced by a factor of at least μ and the number of heavy edges in M is not changed.

Step-(2). Set $I = I_{\leq t-1}$. Note that $|W_{\leq t-1}| = f(Y_{\leq t-1}, I) = f(Y_{\leq t-1}, I_{\leq t-1})$ still holds.

Step-(3). Set $l = t$ and repeat the collapse if possible. Remove all unblocked edges in X_t (since $|Y_t|$ decreases). For each removed unblocked edge e, include it in I if $f(Y_{\leq t-1}, X_{\leq t} \cup I + e) > f(Y_{\leq t-1}, X_{\leq t} \cup I)$.

4.4 Invariants and Properties

▶ **Fact 13** (Key Invariant). *Since the construction of L_t (until L_{t-1} is collapsed), $f(Y_{\leq t-1}, X_{\leq t} \cup I)$ does not decrease and is always no less than $|X_{\leq t}|$.*

Proof. We prove by induction on $t \geq 1$. Consider the base case when $t = 1$. The statement trivially holds when L_t is just constructed and when $|X_t \cup I|$ increases. Suppose in some iteration $|X_t \cup I|$ decreases, then it must be because Y_t is collapsed, in which case $f(Y_{\leq t-1}, X_{\leq t} \cup I)$ does not change due to the update rule of step-(3).

Now assume the statement is true for t and consider $t + 1$.

When L_{t+1} is built we have $f(Y_{\leq t}, X_{\leq t+1} \cup I) \geq f(Y_{\leq t-1}, X_{\leq t} \cup I \cup X_{t+1}) = f(Y_{\leq t-1}, X_{\leq t} \cup I) + |X_{t+1}| \geq |X_{\leq t+1}|$. Since $|X_i|$ does not increase afterwards for all $i \leq t + 1$, applying the same argument to L_{t+1} as above yields the fact. ◀

▶ **Lemma 14** (Exponential Growth). *Let $r = \max\{\lceil \frac{k}{9} \rceil, \lceil \frac{k-10}{3+2\sqrt{2}} \rceil\}$, if $T \leq \mathsf{OPT}$, then for all $i \in [l]$ we have $|Y_i| \geq \mu^2 |Y_{\leq i-1}|$, which implies $l = O(\frac{1}{\mu^2} \log n)$.*

Proof Sketch. suppose $|Y_t| < \mu^2 |Y_{\leq t-1}|$, then by Fact 13 we can show that $|X_{\leq t}| \leq f(Y_{\leq t-1}, X_{\leq t} \cup I) < (\frac{r}{p-r+1} + 2\mu)|Y_{\leq t-1}|$ (when μ is sufficiently small). Hence at the moment when there is no more addable edge that can be included into $X_{\leq t}$, we can show that the total number of items agents reachable from $A(Y_{\leq t-1})$ are interested in are not enough to achieve $T \leq \mathsf{OPT}$. Please refer to our full version for the complete proof. ◀

Proof of Theorem 6. For any T and $k = \lceil \frac{T}{\epsilon} \rceil$, the algorithm tries to compute an $r\epsilon$-allocation, for integer r as large as possible, by enumerating all possible values of p between r and k. For any fixed r and p, we try to augment the partial matching M that matches each agent with either a heavy item or r light items. Hence it suffices to show that the algorithm

terminates in polynomial time for augmenting the size of M by one. Since each iteration can be done in polynomial time, it suffices to bound the number of iterations by $\mathsf{poly}(n)$. The approximation ratio will be the maximum of $\frac{k}{r}$, over all $T \leq \mathsf{OPT}$.

By Lemma 14 and the definition of collapsible, we know that after each iteration, either (if no collapse) a new layer with $|Y_{l+1}| \geq \mu^2 |Y_{\leq l}|$ is constructed, or some $|Y_t|$ is reduced to at most $(1-\mu)|Y_t|$ while Y_i are unchanged, for all $i < t$. Let $s_i = \lfloor \log_{\frac{1}{1-\mu}} \frac{|Y_i|}{\mu^{2i}} \rfloor$ and $\mathbf{s} = (s_1, s_2, \ldots, s_l, \infty)$ be the signature, then we have: (1) it is lexicographically decreasing across all iterations: if there is no collapse, then some layer is newly constructed and hence $\mathbf{s}$ decreases; otherwise let L_t be the last layer that is collapsed and $|Y_t|$ be the size of Y_t before it is collapse: we know that at the end of the iteration s_i is not changed for all $i < t$ while $s_t \leq \lfloor \log_{\frac{1}{1-\mu}} \frac{(1-\mu)|Y_i|}{\mu^{2i}} \rfloor = \lfloor \log_{\frac{1}{1-\mu}} \frac{|Y_i|}{\mu^{2i}} \rfloor - 1$ is decreased by at least one, which also means $\mathbf{s}$ decreases; (2) its coordinates are not decreasing: for all $i \in [l-1]$ we have $s_{i+1} = \lfloor \log_{\frac{1}{1-\mu}} \frac{|Y_{i+1}|}{\mu^{2i+2}} \rfloor \geq \lfloor \log_{\frac{1}{1-\mu}} \frac{|Y_{\leq i}|}{\mu^{2i}} \rfloor \geq s_i$. Since we have $l = O(\log n)$ and $s_i = O(\log n)$ for all $i \in [l]$, the total number of iterations (signatures) is at most $2^{O(\log n)} = \mathsf{poly}(n)$.

Approximation ratio: When $k \leq 9$, then a trivial 9-approximation can be achieved by a ϵ-allocation (maximum matching). By the proof of Lemma 14, the approximation ratio $\frac{k}{r}$ is always at most 9 and tends to $3 + 2\sqrt{2} \approx 5.83$ as $\epsilon \to 0$. ◀

References

1. Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In Piotr Indyk, editor, *SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1357–1372. SIAM, 2015. doi:10.1137/1.9781611973730.90.
2. Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24, 2012. doi:10.1145/2229163.2229168.
3. Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In David S. Johnson and Uriel Feige, editors, *STOC 2007, San Diego, California, USA, June 11-13, 2007*, pages 114–121. ACM, 2007. doi:10.1145/1250790.1250808.
4. Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In Jon M. Kleinberg, editor, *STOC 2006, Seattle, WA, USA, May 21-23, 2006*, pages 31–40. ACM, 2006. doi:10.1145/1132516.1132522.
5. Ivona Bezáková and Varsha Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005. doi:10.1145/1120680.1120683.
6. Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 107–116. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.51.
7. Deeparnab Chakrabarty, Sanjeev Khanna, and Shi Li. On (1, epsilon)-restricted assignment makespan minimization. In Piotr Indyk, editor, *SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1087–1101. SIAM, 2015. doi:10.1137/1.9781611973730.73.
8. Uriel Feige. On allocations that maximize fairness. In Shang-Hua Teng, editor, *SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 287–293. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347114.
9. Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab S. Mirrokni. Approximating submodular functions everywhere. In Claire Mathieu, editor, *SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 535–544. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496829.

10 Daniel Golovin. Max-min fair allocation of indivisible goods. Technical report, Carnegie Mellon University, 2005.

11 Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the lovasz local lemma. In *FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 397–406. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.45.

12 Penny E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995. doi:10.1007/BF01793010.

13 Subhash Khot and Ashok Kumar Ponnuswami. Approximation algorithms for the max-min allocation problem. In *APPROX-RANDOM*, volume 4627 of *Lecture Notes in Computer Science*, pages 204–217. Springer, 2007.

14 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990. doi:10.1007/BF01585745.

15 Lukas Polacek and Ola Svensson. Quasi-polynomial local search for restricted max-min fair allocation. In *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 726–737. Springer, 2012.

16 Ola Svensson. Santa claus schedules jobs on unrelated machines. In Lance Fortnow and Salil P. Vadhan, editors, *STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 617–626. ACM, 2011. doi:10.1145/1993636.1993718.