An Approximation Algorithm for Max-Min Fair Allocation of Indivisible Goods

Arash Asadpour Amin Saberi *

Abstract

In this paper, we give the first approximation algorithm for the problem of max-min fair allocation of indivisible goods. The approximation ratio of our algorithm is $\frac{1}{\sqrt{k}\log^3 k}$. As a part of our algorithm, we design an iterative method for rounding a fractional matching on a tree which might be of independent interest.

*Department of Management Science and Engineering, Stanford University, Stanford, CA 94305. Email:{asadpour,saberi}@stanford.edu.

1 Introduction

Fair division, also known as the cake cutting problem has been studied extensively, mostly with a measure theoretic or combinatorial perspective, in mathematics literature since 1950's [16]. More recently this problem has been brought to the attention of computer science community where the emphasis is more on designing efficient algorithms [14, 11, 12]. Moving beyond the metaphor of cake, the focus of most of these works is on practical cases where the goods are less divisible. An interested reader can consult [5] for a long list of applications of these problems.

In this paper, we study the problem of max-min fair allocation of indivisible goods. In our setting, we are allocating m goods to k persons. Each person i has a non-negative integer valuation u_{ij} for good j. We assume that the valuation functions are linear, i.e. $u_{i,C} = \sum_{j \in C} u_{ij}$ for any set C of goods. The goal is to allocate each good to a person such that the least happy person is as happy as possible. More formally, we want to maximize $\min_i u_{i,C_i}$, where C_i is the set of goods allocated to person i.

This problem is very similar in nature to job scheduling problems especially minimum makespan scheduling [13]. However, applications of techniques such as rounding the assignment LP [4, 10] or local search [9] has not yielded algorithms with an approximation ratio better than $O(k)$ for this problem. The most notable progress was made by Bansal and Sviridenko [3] who gave an $O(\log\log k/\log\log\log k)$-approximation schema for the special case of "restricted assignment" in which for all i, j we have $u_{ij} \in \{0, u_j\}$. Recently, Feige [7] showed that the integrality gap of the LP used in [3] is constant.

In this work we introduce the first non-trivial approximation algorithm for the max-min fair allocation problem in the general case. Our algorithm finds an allocation in which the utility of every person is at least $\Omega(\frac{1}{\sqrt{k}\log^3 k})$ of the optimum. In [3], it was shown that the integrality gap of the configuration LP for this problem is $O(\frac{1}{\sqrt{k}})$.

Our algorithm is based upon rounding a configuration linear program similar to [3, 8]. In this formulation, there is a variable $x_{i,C}$ for assigning a valid bundle C to a person i. In our case, a valid bundle is either a *matching* bundle consisting of a single good with high value for i or a *flow bundle* that contains several goods with relatively low value for i. Here, we will explain briefly how we round the solution of this LP. The concrete description of our algorithm can be found in section 2.

Consider the bipartite graph defined by the solution of the LP in which the weights of the edges indicate what fraction of a good is allocated to each person. Also consider the two polytopes defined by the fractional assignment of goods to people via matching and flow edges. Both of these polytopes are very well-behaved. However they are defined on the same vertex set and they can not be rounded separately. In fact, the most critical part of the algorithm is to partition the vertices of the graph between the two polytopes. Once we correctly decide which persons should be satisfied with matching edges it is possible to allocate the rest of the goods to the unmatched persons with a small loss.

We will use a randomized method for rounding the matching edges. For every vertex v corresponding to a good or a person, let m_v be the total fraction of matching edges incident to v. We round the fractional matching in such a way that the probability that a vertex v is saturated by the resulting integral matching is m_v. There are many ways to do this. For example, one can write the fractional matching as a convex combination of a few integral matchings and pick one of the integral matchings at random with probability proportional to its weight. However,

such a rounding scheme will impose lots of undesirable dependencies between different vertices and is not appropriate for our goal.

Instead, we show that without loss of generality we can assume that matching edges form a forest. Then we round the edges of the resulting forest iteratively. The description of this part of the algorithm is given in section 3. Our rounding method tries to avoid imposing additional dependencies between vertices. In fact, we can prove that among all the feasible distributions on the matchings of the forest, the *entropy* of the distribution defined by our rounding algorithm is the highest (for the exact statement see 5.1).

The main part of the analysis of our algorithm is to show that after allocating the goods defined by the matching, there will be enough goods left for every unsaturated person. This is done mainly by showing concentration results on the number of elements released in every bundle. We use generalizations of Azuma-Hoeffding inequality on the martingales defined by the process of our algorithm. This part of analysis is tricky because the expected values of the random variables of interest are sometimes smaller than the maximum step size in the evolution of martingale. In section 4.1 we prove our concentration result.

Finally, in section 5.2 we extend our algorithm to find a solution in which more than $1 - O(\frac{k}{\alpha^2 \log k})$ fraction of people receive a bundle with value at least $\frac{\text{OPT}}{\alpha \log^3 k}$, where OPT is the value of optimal solution in the case that we want to satisfy all the people.

2 Description of the Algorithm

Assume that the the value of the optimal solution, namely T, is known by doing a binary search tree. We model the problem of finding such an allocation by a configuration LP simialr to [3]. A *configuration* is a subset of goods. A configuration C is called *valid* for person i if either $u_{i,C} \geq T$ or it contains a single good j with $u_{ij} \geq T/\sqrt{k}\log^3 k$. Let $\mathcal{C}(i,T)$ denote the set of all valid configurations corresponding to person i with respect to T. The configurational LP relaxation of the problem is as following:

$$\begin{aligned} &\forall j: \sum_{C \ni j} \sum_{i} x_{i,C} \leq 1 \\ &\forall i: \sum_{C \in \mathcal{C}(i,T)} x_{i,C} = 1 \\ &\forall i, C: \ x_{i,C} \geq 0 \end{aligned} \tag{1}$$

Using an argument very similar to [8], one can show that if the above LP is feasible, then it is possible to find a fractional allocation that provides a bundle with value $(1-\epsilon)T$ for each person in polynomial time.

It is shown in [3] that the integrality gap of the above LP is $\Omega(\sqrt{k})$. Here we propose an approach for rounding in which each person will receive a bundle with value at least $\frac{T}{\sqrt{k}\log^3 k}$.

We define a weighted bipartite graph G, with the vertex set $A \cup B$ corresponding to persons and goods respectively. There is an edge between vertices corresponding to person $i \in A$ and good $j \in B$ if a configuration C containing j is fractionally assigned to i. Define $\omega_{i,j} = \sum_{C \ni j} x_{i,C}$, i.e. $\omega_{i,j}$ is the fraction of good j that is allocated to person i by the fractional solution of

Algorithm 1 [The Main Algorithm]

1. Do a binary search to guess T, value of the optimal solution. Solve LP (1). Find the set $\mathcal{M}$ and compute m_v and f_v for every vertex in G.
2. Select a random matching from edges in $\mathcal{M}$ using Algorithm 2 such that for every $v \in A \cup B$ the probability that v is saturated by the matching is $m_v = 1 - f_v$.
3. Let every person i who is not matched yet select a bundle C consisting of small goods with probability $x_{i,C}/f_i$ and claim the goods in that bundle (except the ones that are assigned in the previous step).
4. For each good j, select a person i who has claimed j in previous step uniformly at random and assign that good to her.

Figure 1: : The Main Algorithm

LP (1). A good j is *big* good for a person i, if $u_{ij} \geq \frac{T}{\sqrt{k}\log^3 k}$, otherwise, it is a *small* one. The edge $\{i, j\}$ in the former case is called *matching* edge, whereas it is called *flow* edge in the latter. Let $\mathcal{M}$ and $\mathcal{F}$ represent the set of matching and flow edges respectively. For each vertex $v \in A \cup B$, let m_v be the total fraction of the matching edges incident to it. Also define $f_v = 1 - m_v$.

Our algorithm applies a two-stage randomized rounding to the solution of the LP (1). In the first stage we pick a random matching from $\mathcal{M}$, in a way that with probability m_v the vertex v is in the matching. We will explain the details of the matching algorithm in the next section.

In the second stage every remaining person i selects one of her bundles C consisting of small goods with probability proportional to $x_{i,C}$. From that bundle, she will claim the set of goods that are not saturated by the matching. Note that each person i would not be saturated by the matching with probability $1 - m_i = f_i$. Hence, each such person i will select bundle C with probability $x_{i,C}/f_i$.

Later, we will show that with high probability there will be enough goods left in every bundle i.e. the valuation of i for the goods left in the bundle will remain above $T/\sqrt{k}\log^2 k$.

Note that every good might be claimed by several persons. In the last step, we give every good to one of the persons who have claimed that good uniformly at random. Our algorithm is represented in the Figure 1.

3 Finding a Random Matching

It is possible to construct several examples to show that rounding of matching edges should be done with a lot of care. For example, a close inspection of Bansal and Sviridenko's [3] example for the integrality gap shows that there is very little flexibility in terms of the set of vertices that should be saturated in the matching when we round the fractional solution.

We want that each vertex v in the graph is saturated by the matching with probability m_v. As we pointed out before, we also want that our algorithm does not impose unnecessary dependencies between vertices. In [2] Arora et al. propose a method for "mixing" the integral

Algorithm 2 [Picking a Random Matching]

1. Eliminate cycles of $\mathcal{M}$ using the technique mentioned in the proof of Lemma 1.
2. Pick a vertex v that has not been processed yet.
3. For all edge $e \in \mathcal{M}$: $w_e^{\text{NEW}} \leftarrow w_e$.
   (a) Either pick one of edges adjacent to v, namely e, with probability equal to the weight of these edges and:
      i. $w_e^{\text{NEW}} \leftarrow 1$
      ii. For all edges e' adjacent to e: $w_{e'}^{\text{NEW}} \leftarrow 0$.

   or with the remaining probability for all edges e adjacent to v: $w_e^{\text{NEW}} \leftarrow 0$
   (b) For all other edges e' in the same connected component find a path that connects one of the edges whose weight has become zero in previous step to e'. Let this path be $e_0, e_1, ..., e_l$, where $e_0 = e'$.

$$w_{e_0}^{\text{NEW}} \leftarrow \left[1 + (-1)^{l-1} \frac{w_{e_1} w_{e_2} \cdots w_{e_l}}{(1 - w_{e_1})(1 - w_{e_2}) \cdots (1 - w_{e_l})}\right] w_{e_0}. \tag{2}$$

   (c) For all edges $w_e \leftarrow w_e^{\text{NEW}}$. Remove the edges with weight 0.

Figure 2: Picking a Random Matching

matchings to decrease this dependency but it seems that their method is not accurate enough for our purpose. We will propose a different scheme here. Our scheme takes advantage of the fact that without loss of generality, we can assume that $\mathcal{M}$ is a forest. The following lemma, gives a proof for this simplifying observation.

Lemma 1 *It is possible to adjust the weights of the edges in $\mathcal{M}$ such that for every vertex v, $\sum_u \omega_{u,v}$ remains the same, and the set of edges with non-zero weight form a forest.*

Proof: Suppose that there is a cycle in $\mathcal{M}$ with edges $e_1, e_2, \cdots, e_{2a}$. Let e_1 be an edge with minimum weight among all those edges, namely w, then construct a new solution that is the same as this solution on all other edges, but in these $2a$ edges the weight of odd indexed edges is decreased by w and the weight of the others is increased by w. This process does not change the sum of all edges adjacent to each vertex eventually reduces $\mathcal{M}$ to a forest. □

Our algorithm iteratively picks a verex u that has not been processed yet and matches it with one of its neighbors v with probability ω_{uv}^i or none of them with probability $1 - p_u^i$, where ω_{uv}^i is the weight of edge $\{i, j\}$ before taking the step $i + 1$, and $p_u^i = \sum_v \omega_{uv}^i$.

Let $\mathbf{F}_i$ be the σ-field corresponding to the probability distribution that our algorithm imposes over matchings, before step $i + 1$ is taken. The crucial observation about this algorithm is stated in the following.

Lemma 2 *For all edges e' and for all i, $E[w_{e'}^i | \boldsymbol{F}_{i-1}] = w_{e'}^{i-1}$.*

Proof: If the edge e' is adjacent to the vertex v that we select in the step i then with probability $w_{e'}^{i-1}$ we change its weight to one and with probability $w_{e'}^{i-1}$ to zero and the claim holds. Also Also if e' is adjacent to an edge e which itself is adjacent to v, we have

$$E[w_{e'}^i|\mathbf{F}_{i-1}] = 0 \times w_e^{i-1} + \left[1 + \frac{w_e^{i-1}}{1 - w_e^{i-1}}\right] w_{e'} \times (1 - w_e^{i-1}) = w_{e'}^{i-1}$$

For all other cases, if there is not a path with positive weight intermediate edges between e' and the vertex v before this step, then $w_e^i = w_e^{i-1}$ and we are done. Otherwise let $e_0 = e', e_1, \cdots, e_l$ be such a path between e' and vertex v. Thus,

$$\begin{aligned} E[w_{e'}^i|\mathbf{F}_{i-1}] = &\left[1 + (-1)^{l-2} \frac{w_{e_1} w_{e_2} \cdots w_{e_{l-1}}}{(1 - w_{e_1})(1 - w_{e_2}) \cdots (1 - w_{e_{l-1}})}\right] w_{e'}^{i-1} \times w_{e_l}^{i-1} \\ &+ \left[1 + (-1)^{l-1} \frac{w_{e_1} w_{e_2} \cdots w_{e_l}}{(1 - w_{e_1})(1 - w_{e_2}) \cdots (1 - w_{e_l})}\right] w_{e'}^{i-1} \times (1 - w_{e_l}^{i-1}) \\ &= w_{e'}^{i-1} \end{aligned}$$

□

The following corollary is an immediate result of Lemma 2.

Corollary 3 *The probability that a vertex $v \in A \cup B$ is saturated in the matching generated by Algorithm* 2 *is m_v.*

Another important characteristic of our method which is stated in the following lemma.

Lemma 4 *Our algorithm is* Order Independent*; It means that independent from the ordering, it always gives us a matching M with probability $p(M)$ that only depends on the initial weights of edges in the forest and the matching M itself.*

Proof [sketch]: Straightforward calculation shows that in an ordering, swapping two consecutive vertices will not change the distribution of the weights of the edges after the two steps corresponding to these vertices. The Lemma follows by induction. □

4 Analysis

In this part we will prove that the event that our algorithm does not find an integral solution with value $\frac{T}{\sqrt{k}\log^3 k}$ is very rare. We will do that in three steps.

First, we show that for a subset $S \subseteq A$ or $S \subseteq B$, the distribution of the number of vertices saturated from that subset by the random matching is concentrated around its expected value $\sum_{v \in S} m_v$. This is proved in Section 4.1.

Then we will proceed to show that after allocating the goods defined by the matching, there will be enough goods left for every unsaturated person. In particular, the probability that in step 3 of the algorithm a person claims a bundle which the value of its remaining goods is less than $\frac{T}{40\sqrt{k}\log^2 k}$ is very small. This is done in Section 4.2

Finally, we will prove that with high probability a large portion of the goods will be claimed by no more than $O(\log k)$ persons. Therefore, by allocating the goods uniformly at random to the persons who claimed them, we will give every person about $1/\log k$ of the bundle claimed by her.

4.1 Concentration Results for Algorithm 2

Consider a subset $S \subseteq A$. We will prove that the number of vertices in S saturated by the random matching is concentrated around its expected value $\sum_{v \in S} m_v$. The proof for a subset $S \subseteq B$ will be the same. Define the random variable $X^i = \sum_{u \in S} p_u^i$. The next corollary is an immediate result of corollary 3.

Corollary 5 *The random variables $X_0, X_1, \cdots, X_{END} = X$ define a martingale.*

Note that X denotes the number of vertices that will be matched at the end of algorithm 2 from subset S and $EX = X_0$. Observe that the length of jumps can be as big as 1, and our martingale may have k steps. On the other hand, EX_0 might be $\frac{T}{\sqrt{k}}$ which can be as small as $\log^3 k$. Thus, the standard Azuma-Hoeffding's inequality cannot be applied to yield the desired result. Instead, we will use an extension of the Azuma-Hoeffding inequality from [6] (c.f. Section A.1). We will also need the following Lemma which is proved in the appendix.

Lemma 6 *For all i, we have: $|X_i - X_{i-1}| \leq 1$.*

Lemma 7 *If we process vertex $v \in S$ in the step i, then: $Var[X_i|\boldsymbol{F}_{i-1}] \leq 2p_v^{i-1}$.*

Proof: By definition

$$\mathrm{Var}[X_i|\mathbf{F}_{i-1}] = E[(X_i - EX_i)^2|\mathbf{F}_{i-1}] = E[(X_i - X_{i-1})^2|\mathbf{F}_{i-1}] \leq E[|X_i - X_{i-1}||\mathbf{F}_{i-1}]$$

The last inequality is justified by Lemma 6. Now since $E[X_i - X_{i-1}|\mathbf{F}_{i-1}] = 0$, we can write the last term as $2E[X_i - X_{i-1}|\mathbf{F}_{i-1}, X_i > X_{i-1}] \Pr(X_i > X_{i-1})$. By Lemma 18, X_i can be bigger than X_{i-1} only if v is saturated in step i, which happens with probability p_v^{i-1}. Therefore $E[(X_i - X_{i-1})^2|\mathbf{F}_{i-1}] \leq 2p_v^{i-1}$. □

Lemma 8 *For a subset $S \subseteq A$ and a given λ we have $\Pr(X - \mu \geq \lambda) \geq e^{-\frac{\lambda^2}{2(2\mu \log k + \lambda/3)}}$, where $\mu = EX$.*

Proof: Lemma 6 establishes the bounded difference property. Using Lemma 7, we need to bound the sum of the variances in terms of μ. By the order independence proved in Lemma 4, the probability distribution of the solution of Algorithm 2 does not change when we change the order in which the vertices are processed. We will choose an ordering that is more convenient for our analysis. In our ordering:

- The vertices in S are processed before other vertices.
- In each step $i, 1 \leq i \leq |S|$, we pick a vertex v, with minimum p_v^i among all the vertices that have not been processed yet, and process it. We show this vertex by v_i.

By Lemma 18, $\sum_{u\in S-\{v\}} p_u^i \le \sum_{u\in S} p_u^{i-1}$. This implies that the sum of p_u^i's for all the vertices in S that are not processed yet, will not exceed $X_0 = \mu$. Therefore $p_{v_i}^{i-1} \le \mu/(|S| - i + 1), 1 \le i \le |S|$. So we have:

$$\sum_{i=1}^{|S|} \mathrm{Var}[X_i|\mathbf{F}_{i-1}] \le \sum_{i=1}^{|S|} 2p_{v_i^{i-1}} \le 2\sum_{i=1}^{|S|} \mu/(|S| - i + 1) \le 2\mu \log k.$$

Plugging this inequality in Theroem 15 completes the proof. □

Corollary 9 *For a subset $S \subseteq B$ and a given λ, let Y be a random variable denoting the number of vertices in S that are not saturated by Algorithm 2. Then $\Pr(Y - \mu \ge \lambda) \ge e^{-\frac{\lambda^2}{2(2\mu\log(k+|S|)+\lambda/3)}}$, where $\mu = EY$.*

Proof: For each vertex $v \in S$ add a new dummy person and connect it to v with weight $1 - m_v$. The result immediately follows by applying Lemma 8 on set S consisting of all dummy vertices. □

4.2 Failing Bundles

Now we can study the process of rounding flow edges. As we stated in the algorithm, each person i who has not received a big good in Algorithm 2 will select one of her bundles, say C, with probability $x_{i,C}/f_i$. The probability that a person i does not receive a good in Algorithm 2, is $1 - m_i \le f_i$ and the probability that she selects bundle C is at most $x_{i,C}$.

We call the pair (i, C) a *failing pair* if person i is not saturated at the end of Algorithm 2 and her valuation in the set of goods left in C after the matching is less than $T/40\sqrt{k}\log^2 k$.

First, we will prove that the valuation of a person i in what will be left in a bundle C after Algorithm 2 is not much less than its expected value $R_i(C) = \sum_{j\in C} f_j u_{ij}$. Lemma 10 shows that if $R_i(C) > T/\sqrt{k}\log k$, then the probability that (i, C) is a failing pair is very small. Then, in Lemma 12 we prove that $\sum x_{i,C}$ of pairs with small $R_i(C)$ is $o(1)$.

Lemma 10 *If $R_i(C) > T/\sqrt{k}$, then the probability that (i, C) is a failing pair is at most $e^{-\log^2 k/40}$.*

Proof: We have to consider two cases. If C consists of many goods for which i has small valuation, i.e. if $\sum_{j\in C, u_{ij}\le T/k^3} u_{ij} f_j \ge T/2\sqrt{k}$ then (i, C) can not be a failing pair at all. This is because the number of goods saturated in a matching is at most k. Therefore, the total valuation of person i for the goods lost to the matching from this interval is at most T/k^2. Her valuation for the rest of the goods is at at least $T/2\sqrt{k} - T/k^2 \ge T/\sqrt{k}\log k$.

Now suppose $\sum_{j\in C, u_{ij}\le T/k^3} u_{ij} f_j < T/2\sqrt{k}$. We can find an integer $0 \le \delta < 2.5\log k - 3\log\log k$, such that

$$\sum_{T2^\delta/k^3 \le u_{ij} < T2^{\delta+1}/k^3} u_{ij} f_j \ge T/5\sqrt{k}\log k.$$

Let C' be the set of goods j such that $T2^{\delta}/k^3 \leq u_{ij} < T2^{\delta+1}/k^3$. We will have

$$\sum_{j \in C'} f_j \geq \frac{T}{5\sqrt{k}\log k} \times \frac{k^3}{T2^{\delta+1}} = \frac{k^{2.5}}{10 \times 2^{\delta} \log k}.$$

Using Lemma 8 we can see that the probability that less $\frac{k^{2.5}}{20\times 2^{\delta}\log^2 k}$ of goods in C' are not saturated in the matching is at most $e^{-\log^2 k/40}$. To see this note that the RHS of the above inequality is at least $\log^2 k/10$. Now, if more than $\frac{k^{2.5}}{20\times 2^{\delta}\log^2 k}$ goods in C' are left unsaturated, then person i will have a valuation of at least $\frac{k^{2.5}}{20\times 2^{\delta}\log^2 k} \times \frac{2^{\delta}T}{k^3}$ for them which implies that (i, C) is not failing. □

Lemma 11 *If (i, C) is a failing pair then there is a subset $M_i(C) \subseteq C$ such that $|M_i(C)| \geq \sqrt{k}\log^2 k/12$ and for all $j \in M_i(C)$ we have $f_j \leq 24\log k/\sqrt{k}$.*

Proof: We use the same idea applied in the proof of Lemma 10. Using a similar argument as before we can conclude that

$$\sum_{j \in C, u_{ij} \leq T/k^3} u_{ij} \leq T/2.$$

Therefore,

$$\exists r, \frac{T}{k^3} \leq r \leq \frac{T}{2\sqrt{k}\log^3 k} : \sum_{j \in C, r \leq u_{ij} < 2r} u_{ij} \geq \frac{T}{6\log k}.$$

For this bundle C, define C' to be the set of all goods j in C for which $r \leq u_{ij} < 2r$. For each $j \in C', u_{ij} \leq T/\sqrt{k}\log^3 k$. Therefore $|C'| \geq \frac{T/6\log k}{2r} = \sqrt{k}\log^2 k/6$.
$R_i(C)$ and consequently $R_i(M_i(C))$ are at most $T/\sqrt{k}$. It means that $r\sum_{j\in C'} f_j \leq R_i(C') \leq T/\sqrt{k}$, which implies $\sum_{j\in C'} f_j \leq \frac{T}{r\sqrt{k}}$.
But, for half of the goods j in C', f_j is less than twice the average. Therefore,

$$f_j \leq \frac{2T}{r\sqrt{k}|C'|} \leq \frac{24\log k}{\sqrt{k}}. \tag{3}$$

Those goods form the desired set $M_i(C)$. □

Lemma 12 *Let $\mathcal{C}$ be the set of all pairs (i, C) such that $R_i(C) < T/k^{1/2}$. The probability that some person i claims a bundle C such that $(i, C) \in \mathcal{C}$ is $o(1)$.*

Proof: We need to show that $\sum_{(i,C)\in\mathcal{C}} x_{i,C} = o(1)$. Let N be the set of all goods that belong to $M_i(C)$ for a failing pair $(i, C) \in \mathcal{C}$.

$$\sum_{j\in N}(1 - f_j) = \sum_{j\in N} m_j \leq k \implies |N|(1 - \frac{24\log k}{\sqrt{k}}) \leq k \implies |N| < 2k$$

By equation (3) $\sum_{j\in N} f_j \leq |N| \times \frac{24\log k}{\sqrt{k}} \leq 48\sqrt{k}\log k$. For all $j \in N$,

$$\sum_{(i,C)\in\mathcal{C}, M_i(C)\ni j} x_{i,C} \leq f_j.$$

Adding up the above inequality for all j, we derive $\sum_{(i,C)\in\mathcal{C}} |M_i(C)|x_{i,C} \leq \sum_{j\in N} f_j$. We conclude:

$$\sum_{(i,C)\in\mathcal{C}} x_{i,C} \leq \frac{\sum_{j\in N} f_j}{\sqrt{k}\log^2 k/12} \leq 576/\log k = o(1).$$

□

The above lemma shows that with probability $1-o(1)$, every person receives valuation at least $T/40\sqrt{k}\log^2 k$ from the goods left in the bundle she claims.

4.3 Eliminating Conflicts

In this part we deal with the last step of the algorithm, in which we allocate each good uniformly at random to one of the persons who has claimed it. If $(i,j) \in \mathcal{F}$, the probability that i claims a bundle that includes j is $\omega_{i,j}$. Now, $\sum_i \omega_{i,j} \leq 1$. Hence, applying Corollary 8 with probability more than $1-k^{\gamma/4}$ no good j will be claimed by more than $\gamma\log k$ persons.

Now fix a person i. By a simple Markov Inequality argument with probability at list $1-k^{\gamma/4}/2$ no more than a fraction of $2/k^{\gamma/4}$ of the goods in the bundle she has claimed are allocated to more than $\log k$ goods. Therefore the probability that she still has a valuation at least $T/(40\sqrt{k}\log^2 k)\gamma\log k)$ in her selected bundle that no more than $\gamma\log k$ other persons have claimed it, is at least $1-k^{\gamma/4}$.

Choosing $\gamma = 8$, we conclude that with probability $1-o(1)$ this event happens for all persons. Now it is time to assigning each good to one of the persons who claimed it. By applying Chernoff inequality we prove our main result.

Theorem 13 *With probability $1-o(1)$ our algorithm assigns each person a bundle with valuation at least $T/320\sqrt{k}\log^3 k$.*

5 Further Discussion

5.1 Maximum Entropy

In section 3, we saw that without loss of generality, we can assume that $\mathcal{M}$ is a forest. Then we gave an iterative algorithm for rounding the edges of $\mathcal{M}$ to get a matching. Define a distribution D on the possible matchings of $\mathcal{M}$ feasible if and only if for every vertex v, the probability that it is saturated by a matching in D is m_v.

Theorem 14 *The distribution over matchings of $\mathcal{M}$ imposed by the output of Algorithm 2 has the highest entropy among all feasible distributions.*

We give a brief sketch of the proof here. By writing the convex program characterizing the maximum entropy distribution and writing the complementary slackness conditions, one can show that D^* , the distribution with the highest entropy, belongs to an "exponential family". In other words, it is possible to find $Y : E(G) \mapsto \mathbb{R}$ such that the probability of a certain matching M in D^* is proportional to $\Pi_{e\in M}Y(e)$. It is easy to show that in a distribution like

that the events corresponding to two different trees in the forest are independent. Therefore, for a path $e_0, e_1, \cdots, e_k$ in the tree $\Pr[e_0 \in M, e_k \in M | e_1 \notin M] = \Pr[e_0 \in M | e_1 \notin M] \Pr[e_k \in M | e_1 \notin M]$, where M is a matching selected by distribution D^*. It follows from this equation that $\Pr[e_0 \in M | e_k \notin M] = \Pr[e_0 \in M | e_1 \notin M] \Pr[e_1 \notin M | e_k \notin M]$. Now, we can prove by a simple induction on k, that if a distribution D satisfies this property, then

$$\Pr[e_0 \in M | e_k \notin M] = x_{e_0} + (-1)^{k-1} \frac{x_{e_1} \cdots x_{e_k}}{(1 - x_{e_1}) \cdots (1 - x_{e_k})} x_{e_0}$$

where x_e is the probability that e is picked in a matching M selected with respect to D. This is exactly the update rule of our algorithm. Thus, our algorithm yields the maximum entropy probability distribution.

5.2 Extension

In our rounding mechanism, we can improve the guarantee for most people if we allow a small fraction of the people to remain unhappy. More precisely, let OPT be the optimal solution for the max-min fair problem. For any $\alpha > 48 \log k$, we can allocate the goods to people so that everybody except $O(\frac{k}{\alpha^2 \log k})$, receive a bundle whose value is at least $\frac{\text{OPT}}{\alpha \log^3 k}$.

In order to do this we need to redefine the notions of valid configuration, matching/flow bundles, and big/small goods by replacing $\sqrt{k}$ with α. We will use the same rounding method as in the Algorithm 1, but with respect to the new definition of matching and flow edges. The analysis of the algorithm remains the same except that $\sqrt{k}$ will be replaced by α in the extensions of Lemmas 10, 12 and 4.3. In extension of Lemma 12 we can show that the expected number of failing pairs in the third step of our algorithm is of $O(\frac{k}{\alpha^2 \log k})$. Therefore with probability at least 1/2, the actual number of failing pairs will not be more than twice this number.

Acknowledgement: We would like to thank Nikhil Bansal, Uriel Feige, Nicole Immorlica, and Soheil Mohajer for insightful discussions and very useful comments on an earlier version of this paper. We would also like to thank Mohsen Bayati and Andrea Montanari for useful discussions on maximum-entropy distributions.

References

[1] N. Alon and J. H. Spencer. *The probabilistic method*, second edition, John Wiley & Sons, Inc. 2000.

[2] S. Arora, A. Frieze, and H. Kaplan. *A New Rounding Procedure for the Assignment Problem with Applications to Dense Graph Arrangement Problems.*, Math Programming, 2002.

[3] N. Bansal and M. Sviridenko. *The Santa Claus problem*, appeared in Proc. of the ACM Symposium on Theory of Computing (STOC), 2006.

[4] I. Bezakova and V. Dani. *Allocating indivisible goods*, SIGecom Exchanges, 2005.

[5] S. J. Brams and A. D. Taylor. *Fair division: from Cake Cutting to Dispute Resolution*, Cambridge University Press, 1996.

[6] F. Chung and L. Lu. *Coupling online and offline analyses for random power law graphs*, Internet Mathematics, 1, 2004.

[7] U. Feige. *A theorem related to the Santa Claus problem*, manuscript, 2006.

[8] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, *Tight approximation algorithms for maximum general assignment problems*, In 16th ACM-SIAM Symp. on Discrete Algorithms (SODA), 2006.

[9] Lisa Fleicher, N. Immorlica. *Personal Communications.*

[10] D. Golovin. *Max-min fair allocation of indivisible goods*, Technical Report, Carnegie Mellon University, CMU-CS-05-144, 2005.

[11] J. Kleinberg, Y. Rabani, E. Tardos. *Fairness in routing and load balancing*, appeared in the Proc. of Foundations of Computer Science, 1999.

[12] A. Kumar, J. Kleinberg. *Fairness measures for resource allocation*, appeared in the Proc. of Foundations of Computer Science, 2000.

[13] J.K. Lenstra, D.B. Shmoys, and E. Tardos. *Approximation algorithms for scheduling unrelated parallel machines*, Mathematical Programming, Series A, 1993.

[14] R. Lipton, E. Markakis, E. Mossel, and A. Saberi. *On approximately fair allocations for indivisible goods*, ACM Conference on Electronic Commerce (EC), 2004.

[15] J. M. Robertson and W. A. Webb. *Cake-Cutting Algorithms: Be fair if you can*, AK Peters, 1998.

[16] H. Steinhaus. *The problem of fair division*, Econometrica, 1948.

[17] V.V. Vazirani. *Approximation Algorithms*, Springer-Verlag, Spring 2001.

A Proofs

A.1 Extension of Azuma-Hoeffding's inequality

A *filter* $\mathbf{F}$ is an increasing chain of σ-subfields

$$\{0, \Omega\} = \mathbf{F}_0 \subset \mathbf{F}_1 \subset \cdots \subset \mathbf{F}_n = \mathbf{F}.$$

A martingale (obtained form) X is associated with the filter $\mathbf{F}$ and a sequence of random variables $X_0, X_1, \cdots, X_n$ satisfying $X_i = E(X|\mathbf{F}_i)$ and, in particular $X_0 = EX$ and $X_n = X$. The following result can be found in [6].

Theorem 15 *Let X be the martingale associated with a filter $\boldsymbol{F}$ satisfying*

1. *$Var(X_i|\boldsymbol{F}_{i-1}) \leq \sigma_i^2$, where $1 \leq i \leq n$;*
2. *$|X_i - X_{i-1}| \leq M$, for $1 \leq i \leq n$.*

Then, we have

$$\Pr(X - EX \geq \lambda) \leq e^{-\lambda^2/2(\sum_{i=1}^n \sigma_i^2 + M\lambda/3)}. \tag{4}$$

A.2 Some lemmas needed to prove Lemma 6

Suppose vertex $v \in A$ is being processed during the step i of Algorithm 2. The following Lemma bounds the change in p_u^i with respect to p_u^{i-1}.

Lemma 16 *Choose a vertex $u \in A$ in the same connected component as v. Let u' be the first vertex after u in the path from u to v and v' be the last one before v. If v is matched by $\{v', v\}$ step i, then*

$$p_u^i = p_u^{i-1} - A_{uv'}(1 - p_u^{i-1}),$$

otherwise

$$p_u^i = p_u^{i-1} + A_{uv}(1 - p_u^{i-1}).$$

Proof: Let e' be the edge connecting u and u'. Also Let N be the set of other edges adjacent to u. We will prove the first inequality. The second inequality can be proved the same way. Suppose that the edge $\{v', v\}$ is picked in the step i. From the relation 2 in the Algorithm 2 we have the following:

$$w_{e'}^i = (1 - A_{u'v'})w_{e'}^{i-1}, \forall e \in N : w_e^i = (1 + A_{uv'})w_e^{i-1}.$$

Hence,

$$\begin{aligned}
p_u^i &= (1 - A_{u'v'})w_{e'}^{i-1} + \sum_{e \in N}(1 + A_{uv'})w_e^{i-1} \\
&= (1 - A_{u'v'})w_{e'}^{i-1} + (1 + \frac{w_{e'}^{i-1}}{1 - w_{e'}^{i-1}}A_{u'v'}) \sum_{e \in N} w_e^{i-1} \\
&= w_{e'}^{i-1} + \sum_{e \in N} w_e^{i-1} - A_{u'v'}\left(w_{e'}^{i-1} - \frac{w_{e'}^{i-1}\sum_{e \in N} w_e^{i-1}}{w_{e'}^{i-1}}\right) \\
&= p_u^{i-1} - A_{u'v'}\frac{w_{e'}^{i-1}(1 - w_{e'}^{i-1} - \sum_{e \in N} w_e^{i-1})}{1 - w_{e'}^{i-1}} \\
&= p_u^{i-1} - A_{uv'}(1 - p_u^{i-1}).
\end{aligned}$$

□

We will root the tree at v. Let $\hat{j}$ denote the father of j and C_j denote the set of grandchildren of j. For each vertex j, define T_j to be the set of all vertices in A which lie in the subtree rooted at j.

Lemma 17 *For all the vertices $r \neq v \in A$,*

$$\sum_{u \in T_r} A_{uv}(1 - p_u^i) \leq A_{rv}(1 - w_{r\hat{r}}).$$

Proof: By induction on the height of vertices. For the leaves the left hand side of the inequality is zero and the right hand side is non-negative. Consider some vertex r with even edge distance from v. By the induction hypothesis, for all the vertices $j \in C_r$, we have

$$\sum_{u \in T_j} A_{uv}(1 - p_u^i) \leq A_{jv}(1 - w_{j\hat{j}}).$$

Now we compute the sum for the subtree rooted at vertex r.

$$\begin{aligned}\sum_{u\in T_r} A_{uv}(1-p_u^i) &\le A_{rv}(1-p_r^i) + \sum_{j\in C_r} A_{jv}(1-w_{j\hat{j}}) \\ &= A_{rv}(1-p_r^i) + A_{rv}\sum_{j\in C_r} A_{rj}(1-w_{j\hat{j}}) \\ &\le A_{rv}(1-p_r^i) + A_{rv}\sum_{j\in C_r} \frac{w_{\hat{j}r}w_{j\hat{j}}}{1-w_{\hat{j}r}} \\ &\le A_{rv}(1-p_r^i) + A_{rv}(p_r^i - w_{r\hat{r}}) \\ &= A_{rv}(1-w_{r\hat{r}}).\end{aligned}$$

Which completes the proof. □

Lemma 18 *Let S be an arbitrary subset of A. Then*

$$\sum_{u\in S-\{v\}} p_u^i \le \sum_{u\in S} p_u^{i-1}.$$

Proof: During step i, the values of vertices in a different connected component as v do not change. Also by Lemma 16, the value p_u for an arbitrary vertex u will not increase (decrease) if the edge adjacent to v in its path to u is (is not) processed in the step i. Therefore it is enough to prove the lemma 18 only for the case where:

- The set S consists of all the vertices with even edge distance in the same connected component that v is, e.g. $S = T_v$.
- The vertex v is not saturated after step i.

We use Lemma 17 for all the vertices $u \in C_v$.

$$\begin{aligned}\sum_{u\in T_v-\{v\}} A_{uv}(1-p_u^{i-1}) &\le \sum_{j\in C_v} A_{jv}(1-w_{j\hat{j}}^{i-1}) \\ &\le \sum_{j\in C_v} \frac{w_{\hat{j}v}^{i-1}w_{j\hat{j}}^{i-1}}{1-w_{\hat{j}v}^{i-1}} \\ &= p_v^{i-1}.\end{aligned}$$

It is sufficient to look at the case where $S = T_v$ and the vertex v is not saturated.

$$\begin{aligned}\sum_{u\in S-\{v\}} p_u^i &= \sum_{u\in S-\{v\}} \left(p_u^{i-1} + A_{uv}(1-p_u^{i-1})\right) \\ &= \sum_{u\in S} p_u^{i-1} - p_v^{i-1} + \sum_{u\in S-\{v\}} A_{uv}(1-p_u^{i-1}).\end{aligned}$$

Plugging in inequality 5 we conclude that

$$\sum_{u\in S-\{v\}} p_u^i \le \sum_{u\in S} p_u^{i-1}.$$

□

A.3 Proof of Lemma 6

Proof: The inequality $X_i \leq X_{i-1} + 1$ is implied from Lemma 18 and the fact that $p_v^i \leq 1$. Now, we prove $X_i \geq X_{i-1} - 1$. Note that if vertex v is not saturated then by Lemma 16, for all $u \neq v$, $p_u^i \geq p_u^{i-1}$ so $X_i \geq X_{i-1} - p_v^{i-1} \geq X_{i-1} - 1$. Furthermore, if v is matched to a vertex v', then for all vertices $u \notin T_{v'}$, $p_u^i \geq p_u^{i-1}$. So we need to bound $M = \sum_{u \in T_{v'}} p_u^i - p_u^{i-1}$ by -1 from below.

M can take two different values based on whether or not v is matched to v'. If v is not matched to v',

$$\begin{aligned}
\sum_{u \in T_{v'}} (p_u^i - p_u^{i-1}) &= \sum_{u \in T_{v'}} A_{uv}(1 - p_u^{i-1}) \\
&\leq \sum_{j : \hat{j} = v'} A_{jv}(1 - w_{jv'}^{i-1}) \\
&\leq \sum_{j : \hat{j} = v'} \frac{w_{v'v}^{i-1} w_{jv'}^{i-1}}{1 - w_{v'v}^{i-1}} \\
&= w_{vv'}^{i-1}.
\end{aligned}$$

The first inequality is derived from 17. We know that $EM = 0$. Therefore if v is matched to v', the value of M will be bounded from below by $w_{vv'}^{i-1}(w_{vv'}^{i-1} - 1)/(w_{vv'}^{i-1}) \geq -1$.

□