PTAS for Ordered Instances of Resource Allocation Problems *

Kamyar Khodamoradi[1], Ramesh Krishnamurti[1], Arash Rafiey[1], and Georgios Stamoulis[2]

1 **Simon Fraser University, Burnaby, Canada**
{kka50,ramesh,arashr}@sfu.ca
2 **IDSIA/USI/SUPSI, Manno-Lugano, Switzerland**
georgios@idsia.ch

Abstract

We consider the problem of fair allocation of indivisible goods where we are given a set I of m indivisible resources (items) and a set P of n customers (players) competing for the resources. Each resource $j \in I$ has a same value $v_j > 0$ for a subset of customers interested in j and it has no value for other customers. The goal is to find a feasible allocation of the resources to the interested customers such that in the Max-Min scenario (also known as *Santa Claus problem*) the minimum utility (sum of the resources) received by each of the customers is as high as possible and in the Min-Max case (also known as $R||C_{\max}$ problem), the maximum utility is as low as possible.

In this paper we are interested in instances of the problem that admit a PTAS. These instances are not only of theoretical interest but also have practical applications. For the Max-Min allocation problem, we start with instances of the problem that can be viewed as a *convex bipartite graph*; there exists an ordering of the resources such that each customer is interested (has positive evaluation) in a set of *consecutive* resources and we demonstrate a PTAS. For the Min-Max allocation problem, we obtain a PTAS for instances in which there is an ordering of the customers (machines) and each resource (job) is adjacent to a consecutive set of customers (machines). Next we show that our method for the Max-Min scenario, can be extended to a broader class of bipartite graphs where the resources can be viewed as a tree and each customer is interested in a sub-tree of a bounded number of leaves of this tree (e.g. a sub-path).

1998 ACM Subject Classification G.2.2 Graph Theory, G.1.2 Approximation

Keywords and phrases Approximation Algorithms, Convex Bipartite Graphs, Resource Allocation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.461

1 Introduction and Problem Definition

A bipartite graph $H = (P, I)$ with white vertices P and black vertices I is *convex*, if there is an ordering π of the vertices in I such that the neighborhood of each vertex in P consists of consecutive vertices, i.e., the neighborhood of each vertex in P forms an interval. Convex bipartite graphs are well known for their nice structures and both theoretical and practical properties. Many hard (i.e. **NP**-complete) optimization problems become polynomial-time solvable or even linear-time solvable in convex bipartite graphs while remaining hard for general bipartite graphs [6].

* Fourth author supported by the Swiss National Foundation project 200020-122110/1

We consider the problem of allocating indivisible items (resources) to a set of players (customers) in a convex bipartite graph below.

Problem Description. We are given a convex bipartite graph $H = (P, I)$ together with an ordering π of the vertices of I, where P is a set of n players and I is a set of m items. We consider the problem of allocating the indivisible items from I to the set P. Each player $p \in P$ has a utility function $f_p(j) = v_j > 0$ for each item $j \in [m]$ (v_j is a positive integer). This represents the value of item j for player p. If p is adjacent to item j then its value for p is v_j, otherwise its value is zero. The *goal* is to find a maximum t and a partition $I_1 \cup I_2 \cup \cdots \cup I_n = I$ of the items such that for every $1 \leq j \leq n$, I_j is a subset of items adjacent to player p_j and the items in I_j have a total value at least t in the max-min case and a total value at most t in the min-max case.

The interval case arises naturally in energy production applications where resources (energy) can be assigned and used within a number of successive time steps (i.e. the energy produced at some time step is available only for a limited amount of time corresponding to an interval of time steps) and the goal is a fair allocation of the resources over time, i.e. an allocation that maximizes the minimum accumulated resource we collect at each time step. In other words, we would like to have an allocation that guarantees the energy we collect at each time step is at least t, a pre-specified threshold. See also [19] for some applications in on-line scheduling.

Related work. For the general Max-Min fair allocation problem, where a given item does not necessarily have the same value for each player, no "good" approximation algorithm is known. In [5], by using similar ideas as in [13], an additive ratio of $\max_{i,j} v_{ij}$ is obtained, which can be arbitrarily bad. A stronger LP formulation, the *configuration LP*, is used to obtain a solution at least opt/n in [3]. Subsequently, [2] provided a rounding scheme for this LP to obtain an objective function value no worse than $\mathcal{O}(\frac{\text{opt}}{\sqrt{n}(\log^3 n)})$. In [17], an $\mathcal{O}(\sqrt{\frac{\log\log n}{n \log n}})$ approximation factor, close to the integrality gap of the configuration LP, was shown. In the *restricted* case, where $v_{ij} \in \{0, v_j\}$ for $i \in [n]$ and $j \in [m]$, there is an $\mathcal{O}(\frac{\log\log\log n}{\log\log n})$ factor approximation algorithm [3] for the Max-Min allocation problem. Furthermore, there is a simple $\frac{1}{2}$ inapproximability result for both the restricted case, as well as the general case (where an item does not necessarily have the same value for each player) [5]. Feige proved that the integrality gap of the configuration LP is a constant [8]. In [1] an integrality gap of $\frac{1}{5}$ was shown for the same LP which was later improved to $\frac{1}{4}$. The authors provide a local search heuristic with an approximation guarantee of $\frac{1}{4}$ which is not known to run in polynomial. Later, it was shown in [16] that the local search can be done in $n^{\mathcal{O}(\log n)}$ time. In [10] the authors provided a constructive version of Feige's original nonconstructive argument based on Lovász Local Lemma, thus providing the first constant factor approximation for the restricted Max-Min fair allocation problem. They provide an α-approximation algorithm for some *constant* α where an explicit value of α is not provided. Thus there is still a gap between the $\frac{1}{2}$ inapproximability result and the constant α approximability result in [10].

Several special cases of the Max-Min fair allocation problem have been studied. The case where $v_{ij} \in \{0, 1, \infty\}$ is shown to be hard in [12] and a trade off between running time and approximation guarantee is established. In [4] the authors consider the case in which each item has positive utility for a bounded number of players D, and prove that the problem is as hard as the general case for $D \leq 3$. They also provide a $\frac{1}{2}$ inapproximability result and a $\frac{1}{4}$ approximation algorithm for the *asymmetric* case when $D \leq 2$. The authors also provide a simpler LP formulation for the general problem and devise a polylogarithmic approximation

algorithm that runs in quasipolynomial time. The same result has been obtained in [7], which includes a $\frac{1}{2}$ approximation when $D \leq 2$, thus matching the bound proved in [4]. In [21], the author provides a PTAS for a (very) special case of the problem considered in this paper, namely, when the instance graph of the problem is a complete bipartite graph. In [14] a $\frac{1}{2}$-approximation algorithm was developed for a subclass of instances considered in this paper. See also [18], [15] for other special cases that our results generalize.

The $R||C_{\max}$ problem, as it is known in standard scheduling notation, is an important class of resource allocation problems. In this problem, we have machines (the players) and jobs (the items). Each job can be executed on any machine that belongs to a subset of machines (the subset depends on the job). Furthermore, the time required to process the job depends on the machine it executes on. We seek an assignment of jobs to machines such that the makespan is minimized. For the $R||C_{\max}$ problem, a 2-approximation algorithm based on a characterization of the extreme point solutions of a linear programming relaxation of the problem is given [13]. The authors also provide a $\frac{3}{2}$ inapproximability result. So far, all efforts to improve either of the bounds have failed. In a very recent result [20], it is shown that the restricted version of $R||C_{\max}$ admits an α approximation guarantee for α strictly less than 2. This result is an *estimation* result i.e. it estimates the (optimal) makespan of the restricted $R||C_{\max}$ within a factor of $\alpha = \frac{33}{17} + \epsilon$ for some arbitrary small positive ϵ. In this paper we consider the restricted case of the $R||C_{\max}$ problem where the processing time of each job for the subset of machines is the same.

Outline Of Our Results. Our results can be summarized as follows:

1. We present a PTAS for the restricted Max-Min fair allocation problem when the instance of the problem is a convex bipartite graph (each player sees an interval of items). Notice that this instance of the problem is (strongly) **NP**-complete, as it contains complete bipartite graphs as a special case (each player is adjacent to all the items), which is known to be strongly **NP**-complete [9].
2. We modify our approach for the Min-Max allocation problem to obtain a PTAS for the $R||C_{\max}$ problem when the machines a job can run on are consecutive in some ordering (form an interval).
3. In the Max-Min fair allocation, we show how our techniques can be extended to a bigger class of bipartite graphs. In a convex bipartite graph the items adjacent to a player form an interval or, equivalently, a *path*. In this extension, we consider the case where the items are the vertices of a tree and each player is interested in (has positive evaluation for) items that lie in a *sub-tree with bounded number of leaves* of the tree. We show that our algorithm can be modified to obtain a PTAS, though the run time increases as a polynomial of the number of leaves.

To obtain the PTAS for the instances considered in this paper, we first use scaling to classify the items into small and big items. Because the items adjacent to a player are consecutive, we can construct a solution comprising small items efficiently. We then add the big items to the solution efficiently to construct the total solution.

2 Preprocessing the Input

Consider the convex bipartite graph $H = (P, I)$ together with an ordering π of the vertices in I. For every vertex $p \in P$ let $[\ell_p, r_p]$ be the interval of the items adjacent to p. Based on the ordering π, we define the following ordering on the vertices in P:

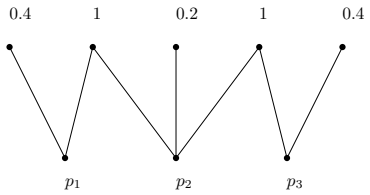

Figure 1 An example of an instance in which Hall's condition is satisfied for $t = 1$ but the optimal solution value is not greater than 0.4.

p is ordered before q whenever $\ell_p < \ell_q$, or $\ell_p = \ell_q$ and $r_p \leq r_q$ (breaking ties arbitrarily). According to ordering π, if $p \in P$ is adjacent to $i \in I$ and $q \in P$ is adjacent to $j \in I$ with $p < q$ and $j < i$ then p is also adjacent to j.

By a *feasible* assignment we mean an assignment such that each item is assigned to exactly one player that has non-zero evaluation for that item.

▶ **Definition 1** (t-assignment)**.** A t-assignment, $t \geq 0$, is a feasible assignment such that every player p receives a set of items $I_p \subseteq [\ell_p, r_p]$ with total value at least t.

Given a particular instance $H = (P, I)$ of the problem, we perform some steps that simplifies the input instance. For a positive integer t, we may assume that the value of each item is at most t. If item j has value $v_j > t$ then we set v_j to t without loss of generality. By a proper scaling, i.e. dividing each value by t, we may assume that the value of each item is in $[0, 1]$. Observe that a t-assignment becomes a 1-assignment. We do a binary search to find the largest value of t for which each player receives a set of items with total value at least t. The binary search is carried out in the interval $[0, \frac{1}{n}\sum_{j\in I} v_j]$ where 0 is an absolute lower bound, and $\frac{1}{n}\sum_{j\in I} v_j$ is an absolute upper bound of the optimal solution respectively.

For a subset $P' \subseteq P$ of players, let $N(P')$ be the union of the set of all neighbors of the players in P'. For an interval $[i, j]$ of the items, let $P[i, j]$ be the set of players whose *entire* neighborhood lies in $[i, j]$: $P[i, j] = \{p \in P : N(p) = [\ell_p, r_p] \subseteq [i, j]\}$. For a subset $I' \subseteq I$ of items, let $v(I')$ denote the sum of the values of all the items in I'. By *private neighborhood* of a player p we mean all the items that are adjacent only to p. We note that in every 1-assignment, for every subset $P' \subseteq P$ of players, the value of the items in its neighborhood should be at least $|P'|$. In other words, $\forall P' \subseteq P : \; v(N(P')) \geq |P'|$. If the value of each item is 1 then this condition is the well known Hall's condition [11], a condition sufficient and necessary for a bipartite graph to have a perfect matching. From now on we refer to the above condition as Hall's condition. Lemma 2 shows that in order to check Hall's condition for H it suffices to check it for every interval of items, and so Hall's condition in our setting becomes Condition (1) below:

$$\forall \;\; [\ell, r] \subseteq [1, m] : \qquad v([\ell, r]) \geq |P[\ell, r]|. \tag{1}$$

▶ **Lemma 2.** *In order to check Hall's condition for H it suffices to verify Condition (1). In other words, it suffices to check Hall's condition for every set of players $P[\ell, r]$, $[\ell, r] \subseteq [1, m]$.*

Note that the value $\frac{v([1,m])}{n}$ is an upper bound on the optimal value. In Figure 1 Hall's condition is satisfied but the optimal value is 0.4. This shows the integrality gap of the ILP formulation for the problem is more than 2. Thus, a different approach is required to get even a $\frac{1}{2}$ approximate solution.

For any integer $k \geq 3$, we let $\frac{1}{k}$ be the error parameter. For each instance for which there is an optimal 1-assignment, we seek an assignment such that each player receives a set of items with total value at least $1 - \frac{1}{k}$, $k \geq 3$. We call an item *small* if its value is less than $\frac{1}{k}$, otherwise it is considered a *big* item. We further round the values of the big items as follows. If v_j (the value of item j) is in the interval $[\frac{1}{k}(1+\frac{1}{k})^i, \frac{1}{k}(1+\frac{1}{k})^{i+1})$ then it is replaced by $\frac{1}{k}(1+\frac{1}{k})^{i+1}$. After the rounding, there are at most $K = \lceil \frac{\log k}{\log(1+\frac{1}{k})} \rceil$ distinct values more than $\frac{1}{k}$. Using straightforward calculus, one can show that K is no more than $k^{1.4}$. For i, $1 \leq i \leq K$ let $q_{i+1} = \frac{1}{k}(1+\frac{1}{k})^{i+1}$. For subset I' of I let $v_s(I')$ denote the value of the small items in I'.

In what follows let $p_1, p_2, \ldots, p_n$ be the ordering of the players and let m be the number of items in H. We also assume the following because it is a necessary condition for having an optimal 1-assignment.

Assumption: *A 1-assignment (an optimal 1-assignment) assigns to each player p_i a set of big items with total value $1 - w_i$ for some "deficit" w_i, $0 \leq w_i \leq 1$, and produces an instance H' of the problem for which Hall's condition is satisfied, i.e. for every interval $[\ell, r]$ of items, $v_s([\ell, r]) \geq \sum_{p_i \in P[\ell,r]} w_i$, that is, the deficit w_i of player p_i must be compensated for with small items.*

3 Structural Properties and the Algorithm

We start with a crucial lemma that will constitute the core of our algorithms. Intuitively, the lemma says that if a 1-assignment exists, then there exists another "almost" 1-assignment with a very particular structure.

▶ **Lemma 3.** *Suppose there exists an optimal 1-assignment for H in which player p_n (last player) receives a set S of items from $N(p_n)$, containing α_i, $1 \leq i \leq K$ big items with value q_i and a set of small items with total value at least $\frac{\alpha_0}{k}$ and less than $\frac{\alpha_0+1}{k}$ such that $v(S) \geq 1$. Then we obtain (in polynomial time) an assignment such that:*

1. *for every $i \geq 1$ the items with value q_i are the rightmost ones in the neighborhood of p_n.*
2. *p_n gets a set of consecutive small items from right to left (in the ordering) of the interval $N(p_n)$ with value at least $\frac{\alpha_0-1}{k}$.*
3. *The existence of a 1-assignment for the rest of H is preserved.*

Proof. *Proof of 1.* Suppose there are two big items x_1, x_2, $x_1 < x_2$, with the same value in the neighborhood of p_n such that $x_1 \in S$ and $x_2 \notin S$. Then item x_2 is either assigned to some player $p_i < p_n$ by the optimal solution or it is not assigned to any player. If x_2 is not assigned to any player by the optimal 1-assignment then we can include it instead of x_1. If x_2 is assigned to p_i in the optimal 1-assignment then x_1 is also adjacent to p_i by the ordering property and we can assign x_1 to p_i and x_2 to p_n.

Proof of 2. We note that we may look at the optimal 1-assignment as follows. The optimal 1-assignment assigns a set of big items to each player p_i with total value $1 - w_i$, $0 \leq w_i \leq 1$ in the first step. After this step, we have an instance of the fair allocation problem where each player p_i, $1 \leq i \leq n$ is allocated a set of small items with total value at least w_i. Because the solution consists of only small items we have $\frac{d_i}{k} \leq w_i \leq \frac{d_i+1}{k}$ for some integer d_i, $0 \leq d_i \leq k-1$. Since there is an optimal 1-assignment, Hall's condition is satisfied for each set of players. Also by Lemma 2, Hall's condition needs to be verified only for each interval of items. For every interval $[\ell, r]$ of the items, we have Condition (2) below:

$$v([\ell, r]) \geq \sum_{p_i \in P[\ell,r]} w_i \tag{2}$$

Let $S(p_n)$ be the set of items obtained as follows. Start from r_{p_n}, the *last* item in the neighborhood of p_n, and add the small items one by one from right to left to set $S(p_n)$, as long as $v(S(p_n)) < w_n - \frac{1}{k}$. Then, we add the next rightmost small item to the set $S(p_n)$ as well (so $v(S(p_n)) \geq w_n - \frac{1}{k}$). Let $\ell_s(p_n)$ be the index (according to the ordering) of the leftmost item added to $S(p_n)$. Note that we may need to add all of the small items to $S(p_n)$. Observe that $w_n - \frac{1}{k} \leq v(S(p_n)) < w_n$ since the last item added to $S(p_n)$ has value less than $\frac{1}{k}$. By assigning $S(p_n)$ to p_n and removing it from H, Condition (2) is still satisfied for each interval of items in the rest of the graph. Observe that since Hall's condition is satisfied, $v_s(N(p_n)) \geq w_n$. On the other hand, $v(S(p_n)) < w_n$. We assign $S(p_n)$ to p_n, and we observe that the items in $S(p_n)$ are consecutive. We will show that for the rest of the players and items, Hall's condition is still satisfied.

Proof of 3. Let $H' = H \setminus (S(p_n) \cup \{p_n\})$ be the reduced instance we derive after assigning items in $S(p_n)$ to player p_n. Note that the neighborhood of each player in H' is an interval. Consider an interval $[\ell, r]_{H'}$ in H' such that $P_{H'}[\ell, r] \neq \emptyset$ in H'. If $[\ell, r]_{H'} \cup S(p_n)$ is not an interval in H then Hall's condition is satisfied for $[\ell, r]_{H'}$ as otherwise $[\ell, r]_{H'} = [\ell, r]_H$ and Hall's condition would not be satisfied in H. So we assume $[\ell, r]_{H'} \cup S(p_n)$ forms an interval in H. Consider the set of items $[\ell, r]_{H'} \cup S(p_n)$ in H (an interval in H). We note that $S(p_n)$ corresponds to interval $[\ell_s(p_n), r_{p_n}]$ in H. First we notice that $\ell \leq \ell_s(p_n)$ (i.e. ℓ is to the left of $\ell_s(p_n)$). This follows from the ordering of the players based on the left end points of their intervals. Thus we have $[\ell, r]_{H'} \cup S(p_n) = [\ell, r]$. Moreover $P[\ell, r] = P[[\ell, r]_{H'}] \cup \{p_n\}$. Therefore we have $v([\ell, r]_{H'}) + v(S(p_n)) = v[\ell, r] \geq \sum_{p_i \in P[\ell,r]} w_i$. Since $v(S(p_n)) < w_n$, we have $v([\ell, r]_{H'}) \geq \sum_{p_i \in P[[\ell,r]_{H'}]} w_i$. ◀

The Algorithm: We first observe that if an optimal 1-assignment assigns a set of items containing α_i items with value q_i to player p_n then by Lemma 3 we may assume that these α_i big items are the rightmost big items of value q_i in the neighborhood of p_n.

Before we proceed, we need the following definition of the *right-most vectors* (intuitively vectors that satisfy the conditions of Lemma 3).

▶ **Definition 4** (Right-most Ordering)**.** Let $V = (\alpha_0, \alpha_1, \alpha_2, ..., \alpha_K)$ be a vector of non-negative integers. For a given interval of items $[\ell, r] \subseteq [1, m]$ let $\mathcal{S}([\ell, r])$ be all the sets of items $S \subseteq [\ell, r]$ that are consistent with V i.e. $S \in \mathcal{S}([\ell, r])$ if the vector of items that represents S is exactly V. There might be several different sets S in $[\ell, r]$ consistent with V. We say that S is a **right-most** set of items in $[\ell, r]$ if

- for each i, $1 \leq i \leq K$, S contains the rightmost α_i big items with value q_i from $[\ell, r]$.
- the small items in S are the rightmost consecutive small items from $[\ell, r]$.

Observe that such a set S in our setting is **unique**. Moreover, when we say that a vector of non-negative integers V is the right-most for a given interval $[\ell, r]$, we interpret it as the unique $S \in \mathcal{S}([\ell, r])$ with the properties listed above.

At each step i, $1 \leq i \leq n$ of the algorithm we keep track of the right-most vectors of items assigned to the players $p_{n-i+1}, p_{n-i+2}, \ldots, p_n$ as well as the subgraph left for the rest of the players. We call such a vector an *assigned vector* and there might be several such assigned vectors at step i. Each assigned vector $A_i = (\beta_0, \beta_1, \ldots, \beta_K)$ at step i indicates that all together β_j items of value q_j, $1 \leq j \leq K$ and a set of S' of small items with value $\frac{\beta_0}{k}$ can be assigned to players $p_{n-i+1}, p_{n-i+2}, \ldots, p_n$, i.e. A_i represents an assignment to the players $p_{n-i+1}, p_{n-i+2}, \ldots, p_n$.

In order to keep track of the right-most assigned vectors and subgraphs we construct an $n \times d$ matrix M. Here $d = (K+1)m^{K+1}$ is the number of all possible assigned vectors

that arise from the initial vector $V = (z_0, z_1, z_2, ..., z_K)$ representing all the items (recall that $m = |I|$). Each entry of M contains one bit (which is either 0 or 1) and an $n \times m$ adjacency matrix. In particular, $M[i, j] = 1$ if assigning some right-most vector indexed by j to player p_{n-i+1} makes this vector "active" in the next round (i.e., it potentially can lead to a valid assignment for all players, therefore should be considered). Moreover, with abuse of notation, we say that $M[i, j] = H' \subseteq H$ where H' is the subgraph that arises by ignoring the items from the newly assigned vector j and players after p_{n-i}. Once we consider player p_{n-i+1} we consider a right-most vector $V_i = (\alpha_0, \alpha_1, \ldots, \alpha_K)$ (in the neighborhood of p_{n-i+1}) with value at least $1 - \frac{1}{k}$ that includes all the items in the private neighborhood of p_{n-i+1} (as otherwise they will not be used later). Then we look at an entry $M[i-1, j'] = 1$, where j' represents the right-most vector $(\beta_0, \beta_1, \ldots, \beta_K)$ and we set $M[i, j] = 1$, where j corresponds to vector $V_i = (\alpha_0 + \beta_0, \alpha_1 + \beta_1, \ldots, \alpha_k + \beta_K)$. Moreover $M[i, j] = H' \subseteq H$ where H' is the subgraph that arises by ignoring the items from the current assigned vector and ignoring the players $p_{n-i+1}, p_{n-i+2}, \ldots, p_n$. Note that several possible configurations may set an entry to one. This subgraph is obtained from the set of items corresponding to V_i and the subgraph from $M[i-1, j']$.

Algorithm for Convex Case

At step i (at the beginning $i = 1$):

1. The current player for consideration is p_{n-i+1}. Let $\mathcal{A}_i$ be the current set of all right-most assigned vectors i.e. $\mathcal{A}_i = \{j \in [d] : M[i-1, j] = 1\}$ ($\mathcal{A}_1 = \emptyset$). For each $A \in \mathcal{A}_i$ do:
   - **a.** Consider all the *minimal* right-most vectors $V_i = (\alpha_0, \alpha_1, \alpha_2, ..., \alpha_k)$ representing items from $N(p_{n-i+1})$ in the subgraph induced by the *current* assigned vector A (this subgraph can be simply found by consulting the corresponding entry in the matrix M) such that:
     - $1 - \frac{1}{k} \leq \frac{\alpha_0}{k} + \sum_{j=1}^{j=K} \alpha_j q_j$
     - all the items corresponding to this vector are in the neighborhood of p_{n-i+1}
     - V_i includes all the private neighbors of p_{n-i+1} (items adjacent only to p_{n+1-i})
     - If the value of the items in the private neighborhood of p_{n-i+1} is at least $1 - \frac{1}{k}$ then let V_i be the vector of all the items in the private neighborhood of p_{n-i+1}

     Observe that each vector V_i represents a *unique* set of items since it is a right-most vector.
   - **b.** If there is no such V_i, report NO assignment and exit.
   - **c.** For every such vector V_i (at step i) we consider the assigned vector $\overline{V_i} = A_i + V_i$ (observe that, given V_i, the assigned vector $\overline{V_i}$ is uniquely defined):
     - Set $M[i, j] = 1$ where j is the column corresponding to this assigned vector $\overline{V_i}$.
     - Update the entry in $M[i, j]$ corresponding to the subgraph induced by players $p_1, p_2, \ldots, p_{n-i}$ by using the previous entry of the corresponding subgraph at step $i-1$ and the current set of items in vector V_i (at Step 1 we use the adjacency matrix of H).

     Step (c) above keeps track of the remaining subgraph for the rest of the players.
2. Set $i = i + 1$ and go to (1).
3. Assign the items in the neighborhood of p_1 (corresponding to one of the subgraph remained containing p_1) and trace back M to obtain an assignment for the rest of the players.

In order to retrieve an actual assignment (last step of the algorithm) we proceed as follows: at step n when we consider player p_1 there should be at least one vector of items with value $1 - \frac{1}{k}$ in the neighborhood of p_1). We assign the items that are uniquely defined

by such a vector to player p_1. To continue with the rest of the players we may find useful to include the index j at a step i that caused a particular assigned vector at the next step $i+1$ be set to 1. With this, when we allocate a particular vector of items to player p_{n-i} we know how to trace back a feasible assignment. In other words, whenever we set $M[i,j]=1$ in the body of the algorithm we also store which assigned vector j' from row $i-1$ is responsible for setting $M[i,j]=1$ at step i.

▶ **Lemma 5.** *Let H be an instance of the problem with n players and m items. Suppose there is an optimal 1-assignment for H. Then the Algorithm assigns (in polynomial time) to each player a set of items with value at least $1-\frac{2}{k}$.*

Proof. First by definition of right-most vector, the value of set S corresponding to vector V_i is at least $1-\frac{2}{k}$. This is true because $1-\frac{1}{k} \leq \frac{\alpha_0}{k} + \sum_{j=1}^{j=K} \alpha_j q_j$ and the value of the small items in S is at least $\frac{\alpha_0-1}{k}$.

Second we need to show that the number of assigned vectors A_i at step i is at most $(K+1)m^{K+1}$. According to Item (2) of Lemma 3 we can take the small items consecutively from right to left. This allows us to look at the small items as a number of blocks of size $\frac{1}{k}$ when they are considered from right to left. Therefore we may assume there are $K+1$ types of items resulting in at most m^{K+1} different possible assigned vectors. When the graph induced by players $p_{n-i+1}, p_{n-i+2}, \ldots, p_n$ and their neighborhood is a complete bipartite graph then the number of possible assignments (number of 1's in the row i of M) is bounded by $(K+1)m^{K+2}$. Moreover, since each right-most assigned vector *uniquely* defines a set of items S, this means that at each step the entry of $M[i,j]$ that corresponds to the subgraph induced for the rest of the players $(p_1, \ldots, p_{n-i})$ is unique. So, the size of the matrix M is $\mathcal{O}(nm^{K+2})$ and each entry of M contains an $m \times n$ adjacency matrix.

Note that each assigned vector at step i represents at least one assignment to the players $p_{n-i+1}, p_{n-i+2}, \ldots, p_n$ such that each of them receives at least $1-\frac{2}{k}$. We claim that if we keep track of at most $(K+1)m^{K+2}$ different possible ways of assigning the items to the players $p_n, p_{n-1}, ..., p_{n-i+1}$ then according to Lemma 3 we guarantee the existing of a 1-assignment for the players $p_1, p_2, \ldots, p_{n-i}$ using the remaining items.

Suppose there exists i, $1 \leq i \leq n$, such that there is no vector V_i in Step 1.b. Then we show that there is no optimal 1-assignment. We use induction on i. Note that i is more than 1 as otherwise there are not enough items in the neighborhood of p_n and clearly there is no optimal 1-assignment. We show that $i > 2$. If $i = 2$ then according to the selection of the items in Step (1) for player p_n we include all the private neighbors of p_n, and all the possible vectors V_1 considered for player p_n are right-most. Hence by Lemma 3 the existence of the 1-assignment should be preserved for the rest of the players, a contradiction.

Let $i \geq 3$. At step $i-1$ the algorithm considers a vector V_{i-1} from $N(p_{n-i+2})$, and together with an assigned vector A_{i-2} from row $i-2$ of M, it creates a new entry for row $i-1$. If the algorithm should have recorded some other assignment different from the ones in the entry of M at row $i-1$ then it means some big item x (of value q_j) and not in the items represented by $A_{i-2}+V_{i-1}$ (or a set X of small items with total value $\frac{\beta}{k}$) is assigned to a player p_t, $n-i+2 \leq t$ and some item x' (of value q_j) from the of items represented by $A_{i-2}+V_{i-1}$ (or a set Y of small items with total value $\frac{\beta}{k}$ represented by $A_{i-2}+V_{i-1}$) is assigned to p_{n-i+1}. Note that $x < x'$. However, since p_{n-i+1} is adjacent to x', it is also adjacent to x and hence we can exchange x and x'. In other words, as far as player p_{n-i+1} is concerned, the items from the right-most assigned vector are the ones that can be assigned to the players $p_{n-i+2}, p_{n-i+3}, \ldots, p_n$. ◀

▶ **Theorem 6.** *Let H be an instance of the problem with n players and m items. Then for $k \geq 3$ there exists a $(1 - \frac{3}{k+1})$-approximation algorithm with running time $O(n^2 m^{K+2})$.*

Proof. According to Lemma 5, each player receives a set of items with value at least $1 - \frac{2}{k}$, once we round the value of the items. Because of the rounding, this value should be divided by $1 + \frac{1}{k}$. Therefore each player receives a set of items with value at least $1 - \frac{3}{k+1}$. The size of the matrix M in Lemma 5 is $O(nm^{K+1})$ and each entry of M contains an $m \times n$ adjacency matrix. Therefore the running time of the algorithm is $O(n^2 m^{K+2})$. ◀

4 Min-Max Allocation Problem ($R||C_{\max}$)

Problem Description: We are given a set M of identical machines and a set J of jobs. Each job j has a same processing time p_j on a subset of machines and it has processing time ∞ on the rest of the machines. The goal is to find an assignment of the jobs to the machines, such that the maximum load among all the machines is minimized. Formally, we have a bipartite graph $H = (M, J, E)$ where M is a set of machines and J is a set of jobs, and E denotes the edge set. There is an edge in E between a machine and a job if the job can be executed on that machine. We consider the case where each job can be executed on an interval of machines:

Assumption: *We have an ordering $M_1, M_2, ..., M_n$ of machines such that each job can be executed on consecutive machines (an interval of machines).*

We denote the interval of job J_i by $[\ell_i, r_i]$. We assume that J_i is before J_j, $i < j$ whenever $\ell_i < \ell_j$ or $\ell_i = \ell_j$, $r_i \leq r_j$. We denote this ordering by π. The ordering π has the following property: if M_i is adjacent to J_r, and M_j for $j > i$ is adjacent to J_s, $s < r$ then M_i is also adjacent to J_s. By scaling down the value of the processing time, we may assume that $0 \leq p_i \leq 1$.

Consider the error parameter $\frac{1}{k}$ for an integer $k \geq 2$. The goal is to find an assignment such that each machine receives a set of jobs with total processing time at most $1 + \frac{1}{k}$, $k \geq 2$, when there exists an optimal 1-assignment. We say a job is *small* if its value is less than $\frac{1}{k}$, otherwise it is called a *big* job. Now we further round the values of the jobs as follows. If v_j (the value of item j) is in the interval $[\frac{1}{k}(1 + \frac{1}{k})^i, \frac{1}{k}(1 + \frac{1}{k})^{i+1})$ then it is replaced by $\frac{1}{k}(1 + \frac{1}{k})^i$. Using this method, we obtain at most $K = \lceil \frac{\log k}{\log(1+\frac{1}{k})} \rceil$ distinct values more than $\frac{1}{k}$. For $1 \leq i \leq K$ let $q_i = \frac{1}{k}(1 + \frac{1}{k})^i$.

We use the usual classification of the jobs into big and small, together with rounding step as in the case of Max-Min allocation. For subset J' of jobs let $w(J')$ ($w_s(J')$) be the sum of the processing times of all the jobs (small jobs) in J'. For every subset M' of machines let $\mathcal{J}[M']$ be the set of jobs whose entire neighborhood lies in set M'. A necessary condition for having a maximum load at most 1 is that for every subset M' of machines $w(\mathcal{J}[M']) \leq |M'|$. In order to check this condition, we need to check it for every interval of machines. For interval $[i, j]$ of machines $M_i, M_{i+1}, ..., M_j$, we look at all the jobs that are executed only on machines $M_i, M_{i+1}, \ldots, M_j$ and if the sum of the processing time of all these jobs is greater than $j - i + 1$ then the condition is violated. For interval $[\ell, r]$, $\ell \leq r$, let $\mathcal{J}[\ell, r]$ be the set of jobs that can be executed only on a subset of the machines in this interval. By argument similar to that used in the proof of Lemma 2, Condition 3 is given below. For simplicity we refer to the condition $\forall \ \ [\ell, r] \subseteq [1, n] : \ \ w(\mathcal{J}[\ell, r]) \leq r - \ell + 1$ as Hall's condition.

Assumption: *A 1-assignment (an optimal 1-assignment) is an assignment that assigns to each machine M_i a set of big jobs with total value $1 - w_i$, $0 \leq w_i \leq 1$ and it produces an*

instance H' of the problem for which the Hall's condition (with respect to the small jobs) is satisfied, i.e. for every interval $[\ell, r]$ of machines, $v_s(\mathcal{J}[\ell, r]) \leq \sum_{i=\ell}^{i=r} w_i$.

Let $N_0[\ell, c]$ be an ordered set of small jobs in the neighborhood of M_ℓ, obtained as follows. We first add all the jobs in $\mathcal{J}[\ell, \ell]$ one by one from left to right (according to ordering π). In step j, $1 \leq j \leq c$, we add to $N_0[\ell, c]$ all the small jobs from $\mathcal{J}[\ell, \ell + j] \setminus \mathcal{J}[\ell, \ell + j - 1]$ one by one from left to right.

Let $N_i[\ell, c]$, $i \geq 1$ be an ordered set of jobs with value q_i obtained as follows. We first add to $N_i[\ell, c]$ all the jobs with value q_i from $\mathcal{J}[\ell, \ell]$ one by one from left to right. In step j, $1 \leq j \leq c$, we add to $N_i[\ell, c]$ all the jobs with value q_i from $\mathcal{J}[\ell, \ell + j] \setminus \mathcal{J}[\ell, \ell + j - 1]$ from left to right.

▶ **Lemma 7.** *Suppose there exists an optimal 1-assignment for H in which machine M_1 (the first machine in the ordering) receives a set S of jobs from $N(M_1)$, containing α_i, $1 \leq i \leq K$, big jobs with value q_i, and a set of small jobs with total value at least $\frac{\alpha_0}{k}$ and less than $\frac{\alpha_0+1}{k}$, such that $v(S) \leq 1$. Then we obtain (in polynomial time) an assignment such that:*

1. *for every $i \geq 1$, the jobs with value q_i are the first α_i's jobs in $N_i[1, c]$ for some $c > 1$.*
2. *M_1 gets a set of consecutive small jobs from $N_0[1, c]$ with value less than $\frac{\alpha_0+2}{k}$ and the existence of a 1-assignment for the rest of H is preserved.*

Let $N(M[\ell, r])$ (a set of jobs) denote the neighborhood of machines $M_\ell, M_{\ell+1}, \ldots, M_r$.

▶ **Definition 8** (Left-most Ordering-$R||C_{\max}$)**.** Let $V = (\alpha_0, \alpha_1, \alpha_2, ..., \alpha_K)$ be a vector of non-negative integers. Let V represent a set S of jobs from $N(M[\ell, r])$ containing α_i, $1 \leq i \leq K$ big jobs of value q_i and a set of small jobs with total value at least $\frac{\alpha_0}{k}$ and at most $\frac{\alpha_0+1}{k}$. We say vector V is a *left-most* vector if:

- for each i, $1 \leq i \leq K$, S contains the first α_i jobs with value q_i from $N_i[\ell, r]$.
- the small jobs in S are the first set of consecutive small jobs from $N_0[\ell, r]$.

Let n be the number of machines and m be the number of jobs and set $d = (K+1)m^{K+1}$. Identical with the Max-Min case, we consider a matrix M with the same properties. The algorithm is similar to the one in the Max-Min case of Section 3 (with the necessary adjustments).

▶ **Lemma 9.** *Suppose there exists an optimal 1-assignment for H. Then there exists a polynomial time assignment that assign all the jobs to the machines without exceeding the maximum load $1 + \frac{2}{k}$.*

▶ **Theorem 10.** *Let H be an instance of the problem with n machines and m jobs. Then for $k \geq 3$ there exists an $(1 + \frac{3}{k+1} + \frac{2}{k^2})$-approximation algorithm with running time $O(nm^{K+2})$.*

5 Max-Min problem when the Items are in a Tree

In this section we consider instances of the problem when the items are the vertices of a tree T and each player is interested in a sub-tree of T with at most d leaves for some constant d. This class of instances contain, the class of convex bipartite instance, as a special case. We notice that if the items are vertices of a tree T and each player is interested in a sub-tree of T then we get the general instances of the problem. To see this we just need to assume T is a star.

Problem Description: We are given a bipartite graph $H = (P, T)$ where P is a set of n players and T is a tree where each node of T is an item. We consider the problem of allocating the indivisible items from I to the set P. Each player $p \in P$ has a utility function $f_p(j) = v_j > 0$ for each item $j \in [m]$ (v_j is a positive integer). This represents the value of item j for player p. If p is adjacent to item j then its value for p is v_j, otherwise its value is zero. For each players p the set of items adjacent to p forms a sub-tree of T with at most d leaves (d is a constant number). The *goal* is to find a maximum t and a partition $T_1 \cup T_2 \cup \cdots \cup T_n = T$ of the items (on T) such that for every j, $1 \leq j \leq n$, T_j is a subset of items adjacent to player p_j and the items in T_j have a total value at least t.

Indexing the tree: The *spine* of T is a longest path in T. Let $SP = v_1, v_2, \ldots, v_q$ be a spine of T. The index of a vertex x in T is the smallest i such that v_i is the closest vertex to x. For two vertices x, y of T we say x is before y, (we write $x \prec y$) if the index of x is less than the index of y. When x, y have the same index i, then $x \prec y$ if x is closer to v_i than y, and no other vertex z in the (x, y)-path is closer to v_i than x (note that the (x, y)-path is unique since T is a tree). In all other cases the order between x and y is arbitrary. The index of subtree P is the index of the vertex with the smallest index among the vertices in P. We say subtree P is before subtree Q and we write $P \prec Q$ if the index of P is less than the index of Q and if P and Q have the same index then the last vertex of P in the ordering $\prec$ lies inside Q.

Ordering the players: We order the players based on their sub-trees, i.e. p is before q if $P \prec Q$ where P, Q are the sub-trees corresponding to p, q.

For subtree T' of T let $P[T']$ denote the set of players whose entire neighborhood lies in T'. Let $p_1, p_2, ..., p_n$ be an ordering of the players. For player p_n, let $\ell_1(p_n), \ell_2(p_n), \ldots, \ell_t(p_n)$, $t \leq d$ be the leaves of $N(p_n)$ where $\ell_i(p_n) \prec \ell_j(p_n)$, $1 \leq i < j \leq t$. Let $x \in N(p_n)$ be the item with the smallest index.

▶ **Definition 11.**
- Let S be a subset of items in $N(p_n)$ with value q_i for an $1 \leq i \leq K$. We say S is *good* if there exist $\beta_1, \beta_2, \ldots, \beta_t$ such that $\sum_{j=1}^{j=t} \beta_j = |S|$ and S comprises of the *last* β_j, (for every $1 \leq j \leq t$) items with value q_i on the path from x to $\ell_j(p_n)$ in the sub-tree $N(p_n)$.
- Let S be a subset of small items in $N(p_n)$. We say S is *good* if there exist items $\ell_1, \ell_2, \ldots, \ell_t$, $\ell_j \preceq \ell_j(p_n)$, $1 \leq j \leq t$ such that S comprises of all the small items on the path from $\ell_j + 1$ to $\ell_j(p_n)$ in the sub-tree $N(p_n)$.

Analogous to Lemma 3 and Theorem 5 we have the Lemma 12 and the Theorem 13 below.

▶ **Lemma 12.** *Suppose there exists an optimal* 1*-assignment for* H *such that player* p_n *receives a set* S *of items from* $N(p_n)$ *where* S *contains* α_i, $1 \leq i \leq K$ *big items with value* q_i *and some small items with total value at least* $\frac{\alpha_0}{k}$ *and less than* $\frac{\alpha_0+1}{k}$ *such that* $v(S) \geq 1$. *Then there exists an assignment in which* p_n *gets a set* S' *of items such that for every* $1 \leq i \leq K$, *there are exactly* α_i *big items with value* q_i *in* S' *forming a good set and the small items in* S' *form a good set with total value at least* $\frac{\alpha_0-1}{k}$. *Moreover, the existence of a* 1*-assignment for the rest of* H *is preserved.*

▶ **Theorem 13.** *Let* H *be an instance of the problem with* n *players and* m *items. Then for* $k \geq 3$ *there exists an* $(1 - \frac{3}{k+1})$*-approximation algorithm with running time* $O(nm^{d \cdot K+2})$.

6 Conclusion and Future Work

In all instances of the problem considered in this paper, a proper ordering has played an important role. However we do not know a dichotomy classification for the instances of the problem that admit a PTAS. We ask for a dichotomy of the following form:

> If H belongs to class X of bipartite graphs then there is a PTAS for Max-Min allocation problem otherwise there is no PTAS.

Acknowledgments. We would like to thank Monaldo Mastrolilli for many useful discussions and for proposing this problem to us.

References

1 A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. In *APPROX-RANDOM*, pages 10–20, 2008.
2 A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC*, pp. 14–121. ACM, 2007.
3 N. Bansal and M. Sviridenko. The santa claus problem. In *STOC*, pages 31–40. ACM, 2006.
4 M. Bateni, M. Charikar, and V. Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *STOC*. ACM, 2009.
5 I. Bezáková and V. Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.
6 A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 1999.
7 D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. In *FOCS*, pages 107–116, 2009.
8 U. Feige. On allocations that maximize fairness. In *SODA*, pp. 287–293. ACM-SIAM, 2008.
9 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
10 B. Haeupler, B. Saha, and A. Srinivasan. New constructive aspects of the lovász local lemma. *J. ACM*, 58(6):28, 2011.
11 P. R. Halmos and H. E. Vaughan. The marriage problem. *American Journal of Mathematics*, pages 214–215, 1950.
12 S. Khot and A. K. Ponnuswami. Approximation algorithms for the max-min allocation problem. In *APPROX-RANDOM*, pages 204–217, 2007.
13 J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *FOCS*, pages 217–224. IEEE, 1987.
14 M. Mastrolilli and G. Stamoulis. Restricted max-min fair allocations with inclusion-free intervals. In J. Gudmundsson, J. Mestre, and T. Viglas, editors, *COCOON*, volume 7434 of *Lecture Notes in Computer Science*, pages 98–108. Springer, 2012.
15 G. Muratore, U. M. Schwarz, and G. J. Woeginger. Parallel machine scheduling with nested job assignment restrictions. *Oper. Res. Lett.*, 38(1):47–50, 2010.
16 L. Polacek and O. Svensson. Quasi-polynomial local search for restricted max-min fair allocation. In *ICALP*, 2012.
17 B. Saha and A. Srinivasan. A new approximation technique for resource-allocation problems. In *ICS*, pages 342–357. Tsinghua University Press, 2010.
18 U. M. Schwarz. A PTAS for scheduling with tree assignment restrictions. *CoRR*, abs/1009.4529, 2010.

19 J. Sgall. Randomized on-line scheduling of parallel jobs. *J. Algorithms*, 21(1):149–175, 1996.

20 O. Svensson. Santa claus schedules jobs on unrelated machines. In *STOC*, pp. 617–626, 2011.

21 G. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operation Research Letters*, 20(4):149–154, 1997.