MaxMin Allocation via Degree Lower-bounded Arborescences *

MohammadHossein Bateni[† ‡]
Computer Science Department
Princeton University
Princeton, NJ 08540
mbateni@cs.princeton.edu

Moses Charikar[†]
Computer Science Department
Princeton University
Princeton, NJ 08540
moses@cs.princeton.edu

Venkatesan Guruswami[§]
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
venkat@cs.washington.edu

ABSTRACT

We consider the problem of MaxMin allocation of indivisible goods. There are m items to be distributed among n players. Each player i has a nonnegative valuation p_{ij} for an item j, and the goal is to allocate items to players so as to maximize the minimum total valuation received by each player. There is a large gap in our understanding of this problem. The best known positive result is an $\tilde{O}(\sqrt{n})$-approximation algorithm, while there is only a factor 2 hardness known. Better algorithms are known for the restricted assignment case where each item has exactly one nonzero value for the players. We study the effect of bounded *degree* for items: each item has a nonzero value for at most D players. We show that essentially the case $D = 3$ is equivalent to the general case, and give a 4-approximation algorithm for $D = 2$.

The current algorithmic results for MaxMin Allocation are based on a complicated LP relaxation called the configuration LP. We present a much simpler LP which is equivalent in power to the configuration LP. We focus on a special case of MaxMin Allocation– a family of instances on which this LP has a polynomially large gap. The technical core of our result for this case comes from an algorithm for an interesting new optimization problem on directed graphs, MaxMinDegree Arborescence, where the goal is to produce an arborescence of large outdegree. We develop an n^{ε}-approximation for this problem that runs in $n^{O(1/\varepsilon)}$ time and obtain a a polylogarithmic approximation that runs in quasipolynomial time, using a lift-and-project inspired LP formulation. In fact, we show that our results imply a rounding algorithm for the relaxations obtained by t rounds of the Sherali-Adams hierarchy applied to a natural LP relaxation of the problem. Roughly speaking, the integrality gap of the relaxation obtained from t rounds of Sherali-Adams is at most $n^{1/t}$. We are able to extend the latter result to a more general class of instances. Along the way, we prove a result about the existence of a perfect matching in a probabilistically pruned graph which may be of independent interest.

Categories and Subject Descriptors

F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems—*Computation on Discrete Structures*

General Terms

Algorithms, Economics, Theory

1. INTRODUCTION

We study the MaxMin allocation problem of indivisible goods. An instance of MaxMin Allocation consists of a set A of n players and a set B of m items, with a non-negative utility value p_{ij} for each player $i \in A$ and item $j \in B$. The utility of a player is an additive function, i.e., if player i is given a subset $B_i \subseteq B$ of items, she gets a utility $\sum_{j \in B_i} p_{ij}$. The goal is to find an allocation of items to players such that the minimum of the utilities of the players, i.e., $\min_{i \in A} \sum_{j \in B_i} p_{ij}$, is maximized. The problem corresponding to the dual objective, i.e., minimizing $\max_{i \in A} \sum_{j \in B_i} p_{ij}$, is the classic Makespan minimization problem for unrelated parallel machines (with items corresponding to jobs and players corresponding to machines). In this work, we consider the MaxMin objective which is natural if we think of items as rewards and look for an equitable, fair allocation of them so that every player surpasses a certain "happiness" threshold. Accordingly, this problem has also been called the "Santa Claus" problem [2, 4]. The MaxMin Allocation problem has received a fair bit of attention lately [2, 3, 4, 6, 9, 10, 11, 13]. We discuss some of the previous work next before mentioning the contributions of this paper.

1.1 Previous work

Lenstra, Shmoys and Tardos [12] give a factor 2 approximation algorithm for the MinMax version, and also show a factor $\frac{3}{2}$ inapproximability result. Closing this gap has been a longstanding open problem. Despite much attention recently, there is a large gap in our understanding of the approximability of the dual objective, MaxMin. The best known algorithm achieves an $\tilde{O}(\sqrt{n})$ approx-

*A full version of this paper is available as a technical report [5].

†Supported by NSF ITR grants CCF-0205594 and CCF-0426582, NSF CAREER award CCF-0237113, MSPA-MCS award 0528414, and NSF expeditions award 0832797.

‡Supported in part by a Gordon Wu Fellowship.

§Currently on leave visiting the Computer Science Department, Carnegie Mellon University. Some of this work was done when the author was a member at the School of Mathematics, Institute for Advanced Study. Research supported in part by a Packard Fellowship.

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

imation ratio in the general case, but only a factor 2 hardness result is known. The objective function for MaxMin is rather fragile, since a small mistake in the allocation might starve a player completely, leading to a very poor approximation ratio. This in some sense captures the high level difficulty faced by algorithms for the MaxMin objective.

Bezáková and Dani [6] achieve an additive approximation for the MaxMin Allocation using the natural LP formulation, guaranteeing a value of at least $\mathrm{OPT} - \max_{ij} p_{ij}$. If the maximum value of some objects are close to the optimum, then this could be a poor guarantee. Besides, they modify the hardness proof of [12] to show a factor 2 inapproximability result for MaxMin Allocation.

Various special cases of the problem have also been studied. The uniform case, when each item has the same value for every player, i.e., $p_{ij} = p_j$ for all i, admits a PTAS [13]. Subsequent works, ending in [1], achieve PTAS for the same setting while considering more general objective functions.

Another important special case has been the *restricted assignment* case considered in [2,4,9], where each item j has some intrinsic value p_j and a player is interested in a subset of items; more formally, $p_{ij} \in \{p_j, 0\}$ for all i, j. Bansal and Sviridenko [4] introduce a new linear programming formulation called the Configuration LP to study the MaxMin Allocationproblem, and by rounding this LP are able to achieve an $O(\log\log n/\log\log\log n)$-approximation for the restricted assignment case. Feige [9] gave a non-constructive proof that the configuration LP has $O(1)$ integrality gap for this special case. Subsequently, Asadpour, Feige and Saberi [2] gave a simpler exponential time 5-approximation algorithm, proving the integrality gap is no more than 5.

Golovin [10] presents an $O(\sqrt{m})$-approximation for the so-called Big Goods/Small Goods case, which is a further special case of the *restricted assignment* case (see below) where item values also need to be either 1 or ∞. For the general problem, Bansal and Sviridenko [4] exhibit an example with integrality gap $\Omega(\sqrt{n})$ for the configuration LP. Asadpour and Saberi [3] show that this is essentially tight, by giving an algorithm to round it with an $O(\sqrt{n}\log^3 n)$ guarantee. This is the best currently known guarantee for the general MaxMin Allocation problem. The best known hardness result remains a factor of 2, and this also holds for the restricted assignment case.

Khot and Ponnuswami [11] also consider a special case where $p_{ij} \in \{0, 1, \infty\}$ for all i, j—this is believed to be as hard as the general case. They establish a tradeoff between runtime and approximation ratio. Specifically, they give an $\frac{n}{\alpha}$-approximation algorithm which runs in time $m^{O(1)}n^{O(\alpha)}$, for any given $\alpha \leq n/2$. They also give a $(2n-1)$-approximation algorithm for a generalization of MaxMin Allocation where utility functions are subadditive.

Ebenlendr, Krčal, and Sgall [8] recently revisited the MinMax objective function in the case when each item has nonzero value for (at most) two players, and further these values are the same for the two players (called the "symmetric" version). This problem can be equivalently viewed as a *Graph Balancing* problem, where one has to orient edges of a weighted undirected graph so that the maximum weighted in-degree is minimized. They are able to improve the approximation factor from 2 down to 1.75, while proving the same 1.5 hardness. In light of this, it is natural to study the effect of the bounded degree items in the MaxMin setting.

Very recently, it was brought to our attention that Chakrabarty, Chuzhoy and Khanna [7] study the MaxMin Allocation problem and give an $O(m^{\varepsilon})$-approximation algorithm that runs in time $O(m^{\frac{1}{\varepsilon}})$ for the general case. As a result they can achieve a polylogarithmic approximation in quasipolynomial time. As explained below, we are able to obtain the same guarantee for an interesting special case which is a natural intermediate point towards solving the general case. Though our works were independent, there are some parallels between our approaches. They also give a $(2+\varepsilon)$-approximation algorithm for an instance of MaxMin Allocation that we will call DegreeTwo. In such instances, each item has nonzero utility for at most two players.

1.2 Contributions

We first study the approximability of the MaxMin problem when each item has a nonzero value for at most B players. We observe that the $D = 3$ case (which we call DegreeThree) is as hard to approximate as the general MaxMin problem. This result also holds for the MinMax objective function. For DegreeTwo (i.e., $D = 2$), we give a factor 4-approximation algorithm for this problem. We stress that our algorithm works in the *asymmetric* case when the item can have distinct nonzero values for the two players interested in it. This is the only positive result for the asymmetric version besides the $\tilde{O}(\sqrt{n})$ approximation algorithm [3] for the general MaxMin problem. We also show a modification of hardness result in [8] gives a factor 2 inapproximability of the degree two case, even for the symmetric case. We should note that, prior to our work, the best approximation factor for the symmetric degree two case is also only 4. This follows from our result, but can also be obtained by customizing the algorithm in [4].

We also present a new LP relaxation for the MaxMin problem. We show that this is equivalent in power to the configuration LP, while being simpler and having a compact formulation. (Thus it can be solved directly, unlike the complicated dual Knapsack methods needed to solve the configuration LP.)

We focus on an interesting special case of the MaxMin problem for which the LP formulations prior to our work have a polynomial integrality gap. For these instances, we devise a polylogarithmic approximation that runs in quasipolynomial time (and an m^{ε} approximation in time $m^{O(1/\varepsilon)}$). The technical core of this result is the development of an algorithm for an interesting new optimization problem on digraphs: MaxMinDegree Arborescence. Given a directed graph with designated sources and sinks, the problem requires us to produce a collection of disjoint arborescences T_i, one rooted at each source such that every non-sink vertex in T_i has outdegree M; the goal is to maximize M. While the problem formulation seems similar to the degree bounded Steiner tree problems that have received much attention recently, the nature of the problem is quite different and requires the development of new techniques. Our algorithm uses a lift-and-project inspired LP relaxation with a dependent rounding procedure that exploits the additional variables and constraints. In fact, we show that the LP we use can be obtained by t rounds of the Sherali-Adams hierarchy applied to a basic LP relaxation for the problem. Our techniques can be interpreted as a rounding algorithm for the relaxation obtained after t rounds of Sherali-Adams. They imply an interesting approximation-rounds tradeoff: the relaxation after t rounds has integrality gap at most $\max(\mathrm{poly}\log(m), m^{O(1/t)})$. This is especially interesting as there has been a lot of recent interest in understanding lift-and-project procedures and very few positive results using lift and project are known. The rounding algorithm itself is similar to a rounding algorithm used for the group Steiner tree problem. However, our analysis is quite different and requires bounding higher moments for a dependent random process on a tree.

1.3 Organization

This extended abstract is organized as follows. In Section 2, we introduce the commonly used LP relaxation for the problem, as well as a new, simpler relaxation we propose. We briefly dis-

cuss low degree instances afterwards. Section 3 gives the factor 4-approximation algorithm for DegreeTwo. In Section 4, we identify a class of instances where the previous LP relaxations have polynomially large gap and devise a polylogarithmic approximation for them. This is the main technical contribution of our work. The core of this result is the development of an algorithm for the MaxMinDegree Arborescence problem which we introduce here. We conclude in Section 5 with a brief overview of extensions to the results in this extended abstract that appear in the full version [5] and future work directions. The appendix has omitted proofs. Due to the lack of space, discussion of the connection to the Sherali-Adams hierarchy is deferred to the full version of the paper [5].

2. PRELIMINARIES

Recall that we are given a set A of players and a set B items and p_{ij} represents the utility of item j for player i. Such an instance is usually considered as a weighted bipartite graph that has players on one side and items on the other. In figures, we will use squares to represent players and circles to depict items.

The natural LP for the problem has an unbounded integrality gap. A more powerful LP, called the *Configuration LP*, was introduced in [4], and has been crucially used in [2, 3, 4, 9].

DEFINITION 1 (CONFIG-LP). *There is a variable x_{iC} for each player i and each valid configuration (aka bundle) C of items. A bundle $C \subseteq B$ is called* valid *for i if and only if $\sum_{j\in C} p_{ij} \geq M$. Let $\mathcal{C}_i$ denote the set of valid bundles for player i. Then, the Config-LP relaxation is as follows.*

(Config-LP)

$$\sum_{i\in A}\sum_{\substack{C\in\mathcal{C}_i\\ C\ni j}} x_{iC} \leq 1 \qquad \forall j \in B \tag{1}$$

$$\sum_{C\in\mathcal{C}_i} x_{iC} = 1 \qquad \forall i \in A \tag{2}$$

$$x_{iC} \geq 0 \qquad \forall i \in A, C \in \mathcal{C}_i. \tag{3}$$

This LP has exponential size, but as noted in [4], the separation oracle needed for the dual is the knapsack problem. So, we can find an approximate (with arbitrary fixed precision) value for this LP in polynomial time. Although the enhancements of Config-LP eliminates some gap examples of the natural LP, the integrality gap still remains as large as $\Theta(\sqrt{n})$ [4].

Note that the LPs studied for this problem are used for feasibility of a guessed value M. When dealing with Config-LP, we usually have a threshold $\tau = M/\lambda$ and we revalue any item of value no less than τ to be M. This way, we might increase the value of the solution by a factor of λ, but we create a gap between "small" and "big" items. One big item is enough to satisfy a player and thus we assume a configuration is either "big" (i.e., it has only one big item) or is "small" which means there are multiple (at least λ) small items in it. Not present in Config-LP but implied are the natural variables $x_{ij} = \sum_{C\ni j} x_{iC}$, which show to what extent an item j is assigned to a player i.

Simpler LP.

We introduce a simplified LP here with which we work. In sharp contrast to Config-LP, it has only polynomially many variables and constraints. Besides making arguments simpler, this gives room to enhance the LP with additional constraints as we do in Section 4. We carry out the same rounding for big items here.

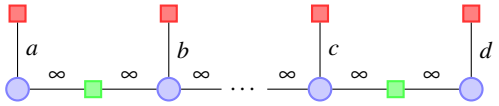

Figure 1: Degree reduction for an item. An item which has degree more than three, can be replaced by a gadget as shown here. The upper row players are the ones which are interested in this item, possibly with different valuations $a, b, \ldots, c, d$. The bottom-row players and items are dummy new ones. It is easy to verify at most one of the original links can be used.

DEFINITION 2 (M-LP). *There is a variable x_{ij} for each player i and each item j. Furthermore, z_i, called the* small usage *of player i, indicates how much small items contribute to the utility of this player. The notion of small/big is with respect to a specific player. Here is the relaxation M-LP:*

(M-LP)

$$\sum_{i\in A} x_{ij} \leq 1 \qquad \forall j \in B, \tag{4}$$

$$\sum_{\substack{j\in B\\ p_{ij}=M}} x_{ij} + z_i \geq 1 \qquad \forall i \in A, \tag{5}$$

$$\sum_{\substack{j\in B\\ p_{ij}<M}} p_{ij}x_{ij} \geq z_i M \qquad \forall i \in A, \tag{6}$$

$$x_{ij} \leq z_i \qquad \forall i \in A, j \in B : p_{ij} < M \tag{7}$$

$$x_{ij}, z_i \geq 0 \qquad \forall i \in A, j \in B. \tag{8}$$

Removing constraints (7) gives the natural LP to which we refer by S-LP. Clearly, any Config-LP solution can be specified as a solution to M-LP. One just needs to use the implied x_{ij} values and appropriate values for z_i. We claim that the algorithms and analyses of previous work using Config-LP can be tailored to use M-LP values instead. Moreover, any solution to M-LP can be turned into a solution of Config-LP of roughly the same value; see Theorem 21. We do not know how to use M-LP (without losing the extra factor two) to get the recent tight result of Chakrabarty et al. [7] for DegreeTwo.

We show that any instance of the problem can be converted into an equivalent instance of DegreeThree.

THEOREM 3 (DEGREE REDUCTION). *Any instance of MaxMin Allocation[1] with n players and m items can be changed into an equivalent instance of DegreeThree that has nm players and nm items.*

PROOF. Take any item of degree d, and replace it by a gadget similar to Figure 1; i.e., make d copies of the item, one for each of its takers (with the associated value), and also add $d-1$ dummy players, one between any two consecutive copies (with value ∞). Each of the dummy players needs to get hold of one of the item copies to satisfy its demand. Exactly one copy will be free to serve its real taker. □

There is a caveat here: The number of players in the degree three instance may be significantly larger and this could affect the approximation ratio.

[1] A similar reduction can be done for MinMax, by replacing ∞ with M (our guess) and noting that we want approximation guarantee better than two.

Knowing that DegreeThree can model the general case of the problem motivates the study of the DegreeTwo special case. S-LP has a large gap for DegreeTwo. Customizing the algorithm of [4], one can obtain a 4-approximation for the symmetric degree two case. Also, implicit in [4, 9] is that the constant degree symmetric case has a constant factor approximation algorithm. More specifically, a restricted assignment instance with items having degree at most $d > 2$, admits a $4d$-approximation algorithm. Yet, in the presence of asymmetry, nothing better than the general $\tilde{O}(\sqrt{n})$-approximation algorithm of [3] was known prior to this work.

We present a factor 2 hardness result for this special case. This is the same as the best-known hardness for the general case.

THEOREM 4 (DEGREETWO HARDNESS). *DegreeTwo cannot be approximated to within a factor better than 2 unless $P = NP$. This also holds if we restrict our instances to be symmetric.*

PROOF. The reduction is from a special 3SAT where each literal appears at most twice. Let the 3SAT instance have variables $v_1, \ldots, v_n$ and clauses $\phi_1, \ldots, \phi_m$. In our MaxMin Allocation instance, we have two players for each variable, one corresponding to v_t and one for $\overline{v_t}$. There is an item of value two shared between them. Each clause has an exclusive item of value 1 and one item shared with any of its literals, with value 1. If some literal occurs l times in our formula, we add $2 - l$ exclusive items of value 1 for it. We claim that the value of the instance is two if and only if the given 3SAT instance is satisfiable.

If it is satisfiable, let the value 2 items be given to the true literals. Each false literal takes its other items to have utility two. Any clause has an exclusive item and at least one shared item, which give it a utility of at least two.

Now, if we have a solution of value at least two, we let the literals which receive big items to be true. False literals take in all their small items. Each clause receives some shared item other besides its exclusive item. So, that should correspond to a true literal.

If the value of the instance is not two, it is at most one, and hence comes the gap of two. □

In the full version of the paper, we discuss a common element in all current hardness reductions of the problem; this is a barrier for proving hardness beyond factor 2.

3. APPROXIMATION ALGORITHM FOR DEGREE 2

To solve an instance of DegreeTwo, we first solve the corresponding M-LP (with big items rounded with $\lambda = 4$). Then, we massage the fractional solution to simplify its structure. Finally, we do the rounding. Next we describe in more details the steps of our algorithm which is presented in Figure 2.

Restructuring the LP solution.

Assume we are given a solution $\{x^*_{ij}, z^*_i\}$ to the M-LP, where all constraints (6) are tight. Otherwise, we first decrease x^*_{ij} values for small items appropriately to obtain this property. We build the new solution x_{ij} as follows. We change all the nonintegral values to $1/2$. Next if any small variable has a value larger than $1/2$, we decrease it to $1/2$. Finally, a player receiving any small item forgoes all its big items. Set $z_i = \frac{2}{M} \sum_{j \in B: p_{ij} < \tau} p_{ij} x_{ij}$ for any player i. The restructured solution has the following properties.

(SP1) Value of a small variable x_{ij} is either zero or $1/2$.

(SP2) Each player uses either big items or small ones.

(SP3) It is a valid M-LP solution of value $M/2$.

Algorithm DEGREETWOSOLVER
Input: *An instance of DegreeTwo.*
Output: *An assignment of value at least $OPT/4$.*

1. Solve M-LP by repeated guessing to get $\{x^*_{ij}, z^*_i\}$. When the guess value is M (revalue any item bigger than $M/4$ to M).
2. Build x_{ij} as follows:
   (a) decrease x^*_{ij} variables as much as possible without violating the constraints;
   (b) let $x_{ij} = x^*_{ij}$;
   (c) change all the nonintegral x_{ij} to $1/2$;
   (d) make sure no $x_{ij} > 1/2$ for small item j; and
   (e) make $x_{ij} = 0$ if item j is big for player i but player i has a nonzero $x_{ij'}$ for a small item j'.
3. Do rotations on big edges to eliminate cycles, and identify T_t's to build the modified instance.
4. Round the fractional solution using LP-ROUND.
5. Deal with the assignment of big items in each T_t.

Figure 2: The algorithm for DegreeTwo.

The truth of (SP1) and (SP2) is immediate. For the last property, note that we have a half-integral solution. There are at most two nonzero variables for each item. If a constraint (4) is violated, then at least one of them has to be one. However, this means that the other one is zero, since it was so in the previous solution. The constraints (6) are good by definition. For the constraints (7), observe that $x_{ij} \leq 1/2$ for small items. If any such constraint is violated, it should be a $z_i < 1/2$ and some $x_{ij} = 1/2$. So in the original solution, x^*_{ij} and thus z^*_i are both positive. Constraints (7) and (6) imply that there are at least M different positive x^*_{ij} for this player i. All these variables have a value $1/2$ in our new solution. Hence $z_i = 1$. This also shows that for a nonzero z^*_i, we have $z_i = 1$ in the new solution, which ensures constraint (5). Now consider constraint (5) for a player with $z^*_i = 0$. Either the player got only one big item or more. In the former case, we do not change the variable value. In the latter, there are at least two nonzero values, each $1/2$, which yield the constraint.

Having the above properties in our fractional solution, we are ready to produce an integer assignment. We are going to use the following result of [6].

THEOREM 5 (ALGORITHM LP-ROUND [6]). *Given a solution to S-LP of value M where the maximum size of an item used is $p^{\max}$, we can round the LP using the algorithm* LP-ROUND *to get a solution of value at least $M - p^{\max}$.*

Although the algorithm cannot give any guarantee when items are large compared to the solution value, it will prove useful in our algorithm.

We build a graph on players and items, where an edge connecting player i and items j has weight x_{ij}. Such an edge is called *big* if item j is big for player i; otherwise, we call it *small*. For now, we ignore the small edges and any player which uses them. Remember that by (SP1)-(SP3), any such player derives enough profit from small items so as not to need any big items. We can perform rotations as in [3] to eliminate cycles in the graph of big edges. Roughly speaking, we pick a small ε and increase the weights of even-indexed edges in the cycle by ε and decrease those of the odd-indexed edges by the same amount. We do this for an appropriate ε

such that the weight of at least one edge drops to zero, when we can remove it. Notice that if an edge gets weight one, in which case, we should also stop, the adjacent edge will have weight zero. Each component of the remaining graph will be a tree T_t. No leaf of a tree T_t can be a player, since otherwise, that would be a player with no small items utility and with only one big item.

We assume, without loss of generality, that any non-leaf item in T_t contributes one unit to the players in the tree. Note that because of degree two bound, no other player can be interested in any such item. If we pick an arbitrary leaf j of T_t and root the tree from it, we can give any item to its child in the tree (because they have degree two unless they are leaves) to satisfy all the players in T_t. All the other leaves will be free. Any solution needs to use exactly one out of the k_t leaf items of each T_t. The usage of small items is at most $1/2$. So, the LP usage of small items in T_t from the outside is at most $k_t/2$.

Now, replace each tree having $k_t \geq 2$ leaves with $\lceil k_t/2 \rceil \leq k_t - 1$ items j_{tl} for $l = 1, \ldots, \lceil k_t/2 \rceil$. These are meant to play the role of the original $k_t - 1$ remaining items. Name the original leaf items in T_t, by $j'_{t,1}, \ldots, j'_{t,k_t}$. Each of them, say j'_{tl}, has exactly one taker outside, say i_{tl}; we know that the usage is exactly $1/2$. The case there is no such player is trivial. Now, we connect j_{tl} to $i_{t,2l-1}$ and $i_{t,2l}$ with the corresponding values they had for $j'_{t,2l-1}$ and $j'_{t,2l}$. The last item $j_{t,\lceil k_t/2 \rceil}$ might get only one player. We let the weights of all these edges be $1/2$. There is a minor technicality here if two players requesting some j_{tl} are the same. Then, there should be a single edge of weight one whose value is the larger of the two values the player has for the two items.

Now, ignoring the big players altogether, we have a feasible S-LP solution of value $M/2$, and we can round it to $M/2 - M/\lambda$ using Theorem 5. After rounding, we know that at most $\lceil k_t/2 \rceil \leq k_t - 1$ items from T_t have been used and so the remaining one can help satisfy the players in T_t with value at least $\tau = M/\lambda$. Choice of $\lambda = 4$ leads to a 4-approximation algorithm for the problem.

4. GOING BEYOND THE CONFIG-LP

In this section, we consider a special case of the MaxMin Allocation problem: The values of items are in the set $\{0, 1, \infty\}$ and each item has value infinity for at most one player. We call the set of such instances InfDegreeOne. If a player i has value 1 for item j, we say that j is a small item for i. If a player i has value ∞ for item j, we say that j is a large item for i.

As we will show, the LP formulation used in the best known results prior to our work for MaxMin Allocation has integrality gap of $\Omega(n^{1/6})$ for such instances.[2] We design an algorithm that achieves an approximation ratio of m^{ε} in time $m^{O(1/\varepsilon)}$ and a polylogarithmic approximation in $m^{O(\log m)}$ time.

In order to do this, we strengthen the LP formulation for MaxMin Allocation; here our simplified M-LP formulation lends itself to incorporating additional constraints. We begin with pointing out the limitations of current LP formulations for these restricted instances.

LEMMA 6. *M-LP (and similarly Config-LP) has a gap of $\Omega(n^{1/6})$ for InfDegreeOne.*

PROOF. Consider the gadget H consisting of $T+1$ players and $2T$ items. Player M_0 is connected to item $J_1, \ldots, J_T$ and has value 1 for each of them. For $1 \leq t \leq T$, item J_t is big (value infinity) for M_t. For each $1 \leq t \leq T$, player M_t has value 1 for $J_{T+1}, \ldots, J_{2T}$. It is easy to note that there's a fractional solution having value 1 ($1/T$ units of small usage) for M_0 and value T for everybody else. In an integral solution, however, if M_0 claims any αT items of his, the corresponding αT players will have only T small items available for them in total. So, the integral solution of the gadget would be at most $\sqrt{T}$.

Let's call the player M_0 of H its distinguished player. Have T^2 copies of H. Call their distinguished players $A_1, \ldots, A_{T^2}$. Attach an item B_t to any of them with value infinity. There is also one player C which has value one for any of these new items B_t. This player can fractionally use $1/T$ from any of these items and so, each A_t needs only $1/T$ from inside its gadget. So, there is a fractional solution of value T. In an integral solution, C needs several of his items. Suppose he claims B_1 along with several others. Then, A_1 needs to be satisfied internally, which gives a solution of value at most $\sqrt{T}$. The size of the instance is $O(T^3)$. The gap is thus $\Omega(n^{1/6})$. □

We further show a hardness result for these instances similar to the previous ones. In fact, we prove this for instances in the intersection of InfDegreeOne and DegreeTwo.

THEOREM 7. *InfDegreeOne cannot be approximated to within better than a factor two unless P = NP.*

PROOF. The proof is similar to that of Theorem 4. The reduction is from a special 3SAT where each literal appears at most twice. Let the 3SAT instance have variables $v_1, \ldots, v_n$ and clauses $\phi_1, \ldots, \phi_m$. In our MaxMin Allocation instance, we have three players for each variable, one corresponding to v_t and one for $\overline{v_t}$; the third one is a dummy. There are two items of value infinity, each shared between the dummy and one of the literals. These items have value one for the dummy, which has another exclusive item of value one. Each clause has an exclusive item of value 1 and one item shared with any of its literals, with value 1 on both sides. If some literal occurs l times in our formula, we add $2 - l$ exclusive items of value 1 for it. We claim that the value of the instance is two if and only if the given 3SAT instance is satisfiable.

If it is satisfiable, let the true literals receive their infinity edges. The dummy player of each variable takes its exclusive item and the infinity item of the false literal. Each false literal takes its noninfinity items to have utility two. Any clause has an exclusive item and at least one item shared with a true literal, which give it a utility of at least two.

Now, suppose we have a solution of value at least two. Notice that the two literals corresponding to a variable cannot both claim their infinity item. Thus, we let the literals which receive infinity items to be true. False literals take in all their noninfinity items. Each clause is given some item other than its exclusive item. So, that should correspond to a true literal.

If the value of the instance is not two, it is at most one, and hence comes the gap of two. □

4.1 The MaxMinDegree Arborescence problem

Our main technical contribution in this section is a solution to a natural new optimization problem on directed graphs which we introduce in this section and relate to the InfDegreeOne problem. In a digraph, any vertex with no incoming edges is called a *source*. Similarly, a vertex with no outgoing edges is called a *sink*.

DEFINITION 8. *An M-pyramid[3] of a digraph G is a subgraph of G with the following properties:*

1. *it is an arborescence, i.e., the in-degree of every vertex is at most 1;*

[2] A similar argument gives a gap of $\Omega(n^{1/4})$ for InfDegreeTwo.

[3] So named because of *pyramid schemes*. See http://en.wikipedia.org/wiki/Pyramid_scheme.

2. *it contains all the sources in G;*
3. *any sink of the arborescence is also a sink in G; and*
4. *the out-degree of any non-sink vertex is at least M.*

Note that an M-pyramid of G need not contain all the vertices of G. Technically, an M-pyramid is a forest of arborescences, but for convenience, we simply say arborescence.

DEFINITION 9. *In an instance of MaxMinDegree Arborescence problem, we are given a digraph G, and the goal is to find an M-pyramid of G that maximizes M.*

We now show how to reduce MaxMin Allocation on instances in InfDegreeOne to MaxMinDegree Arborescence. Consider an instance in InfDegreeOne. We first remove from the instance, all players that have at least two big items. We will assign items to these players later. For the remaining instance, we build a dependency graph as follows.

DEFINITION 10. *The* dependency graph *is a directed graph G. G has a vertex corresponding to every player in the instance, and a vertex corresponding to every item whose utility is only* 0 *or* 1 *(i.e., an item whose infinity-degree is zero). If v is a vertex corresponding to a player i_v, we also associate v with the unique big item of i_v (if it exists). Thus, each vertex in G corresponds to exactly one item. The edges in G are defined as follows: there is an edge from vertex u to v if and only if the player corresponding to u has value 1 for the item corresponding to v.*

If there is an edge between u and v and both vertices correspond to players (i_u and i_v respectively), the player i_u may be assigned the unique big item of i_v, and i_v must then be assigned many small items to compensate. Note that sinks in the dependency graph are exactly the items whose infinity-degree is zero.

LEMMA 11. *Let G be the dependency graph of an InfDegreeOne instance I. Then, the existence of an M-pyramid in G is equivalent to the existence of an assignment of value M for I. In addition, we can do the mapping in polynomial time.*

PROOF. For a player i, let v_i denote the vertex in the dependency graph corresponding to player i. Also, for an item j with infinity-degree zero, let v_j denote the vertex corresponding to item j.

First, assume that we have an M-pyramid in G. We build the assignment for MaxMin Allocation as follows. For a player i, if v_i is present in the M-pyramid, we assign i the items corresponding to the children of v_i in the M-pyramid. If v_i is not present in the M-pyramid, we assign i her big item. Observe that each player gets items worth a value of M, since a player either gets his big item or he receives M items (his children in the graph) each having a value one. We claim that no item is picked by more than one player. It is clear that no two players can claim an item, if both players value it at one. Neither can an item be infinity for both players. Thus if a collision is going to happen, that is between a small item of one player and the big item of another. In this case, by definition of the dependency graph, the latter player should also be present in the arborescence and thus cannot get any big items at the end.

Next, suppose we have a valid assignment of value M. At first, we make sure a player picks his big item if it is available. As long as there is a set of players $P_1, P_2, \ldots, P_k$ such that each P_t gets the big item of P_{t+1} ($P_{k+1} = P_1$, of course), change the assignment such that each P_t gets his big item. Besides, we remove the assignment of a small item to a player, if the player is also assigned a big item. The M-pyramid is built from the assignment of small items. More precisely, the vertices of the M-pyramid consist of vertices v_i for players i who are assigned small items, as well as vertices v_j corresponding to small items j (with infinity-degree zero) that are assigned to players. The sources have to be present in the arborescence, since the corresponding players do not have any big items. Each non-sink vertex clearly has outdegree M. We claim that there are no cycles in the produced subgraph, which means it is a valid M-pyramid, as desired. For the sake of contradiction, assume there is a cycle $P_1, P_2, \ldots, P_k$. Note that all the vertices of the cycle represent players, since item vertices are sinks. By construction of the dependency graph, in the original solution, these players should form a cyclic dependence; i.e., each gets the big item of the next one. However, we removed such chains. □

Finally we show how to handle the players who have more than one big item; these were removed from the instance initially.

LEMMA 12. *Given an instance I of InfDegreeOne, let I' be the instance obtained by removing any player with more than one big item. Given a solution of value M to I', we can obtain a solution of value $\lfloor M/2 \rfloor$ for I in polynomial time.*

PROOF. Let I be the set of all players of infinity degree larger than one that we ignored initially. For all players with infinity degree more than 2, reduce it to exactly 2 by ignoring some items that they value at infinity.

Now consider the existing assignment of small items to players. Let S be the set of players who have been assigned small items. We need to assign infinity items to players in I by taking away some small items from players in S. Build an auxiliary graph G_A with vertices corresponding to players in S. Edges in this graph correspond to players in I. Suppose a player i in I has two items j_1 and j_2 that have value infinity for i. Further suppose that j_1 is currently assigned to player $i_1 \in S$ (as a small item) and j_2 is assigned to player $i_2 \in S$ (as a small item). Then we have an edge (i_1, i_2) in G_A between the vertices corresponding to players i_1 and i_2. This edge is labeled with player i. Notice that i_1 and i_2 may be the same player.

In order to assign items to players in I, we will orient the edges of graph G_A. The orientation corresponds to a reassignment of items as follows: consider an edge (i_1, i_2) corresponding to player $i \in I$. If this edge is oriented from i_1 to i_2, then i is assigned its infinity item that is currently assigned to i_1 and i_2 keeps the other infinity item for i that i_2 is currently assigned. In case $i_1 = i_2$, either orientation implies that player i grabs either of the items and $i_1 = i_2$ loses that item yet keeps the other one. This ensures that every player in I receives an infinity item. In order to ensure that players in S still have a large number of small items assigned to them, we need to ensure that that the out-degree of v is small compared to M for every vertex v in G_A.

Let d_v be the degree of vertex v in G_A. The claim is that there is a way to orient the edges of G_A such that every vertex v has out-degree at most $\lceil d_v/2 \rceil$. Notice if the player v received M_v items in the assignment of I', then her remaining items are at least $M_v - \lceil d_v/2 \rceil \geq \lfloor M_v/2 \rfloor$, since $d_v \leq M_v$. In order to prove this claim, we add a matching of dummy edges between the vertices of odd degree find an Eulerian tour of G_A (with the dummy edges) and orient the edges accordingly. Then the outdegree of a vertex v is at most $\lceil d_v/2 \rceil$. □

4.2 The algorithm for MaxMinDegree Arborescence

Consider a graph G as an instance of MaxMinDegree Arborescence problem. Let T be the optimal M-pyramid for the instance. We show that there is a *nearly optimal* solution T^* to this instance with small depth; the depth of an acyclic directed graph is defined as the number of edges on its longest path. Throughout this section, we use N to denote the number of vertices in G.

LEMMA 13. *If there is an M-pyramid in the instance, then there also exists an $\lfloor M/2 \rfloor$-pyramid of depth no more than $\frac{\log N}{\log \lceil \frac{M}{2} \rceil}$.*

PROOF. The proof is constructive. The vertices of T can be partitioned into levels. Level zero includes all the sources of G. For $i > 0$, level i is the set of vertices in T that have an edge from level $i-1$. For a vertex v in T, let $z(v)$ denote the size of a subtree rooted at v (including v itself).

We build T^* as a union of disjoint trees, one rooted at each source. Consider the component of T rooted at r. Remove the $\lceil M/2 \rceil$ children of r having the largest subgraph sizes. For any remaining child v of r, we have $z(v) \leq z(r)/\lceil M/2 \rceil$; otherwise, since each vertex appears at most once in the tree, the number of descendants of r in the largest $\lceil M/2 \rceil$ subtrees is at least $1 + \lceil M/2 \rceil z(v) > z(r)$ giving a contradiction. Now we recurse for every remaining child of r. For a vertex v of level l that is not pruned during this process, we have $z(v) \leq z(r)/\lceil M/2 \rceil^l$. Since $z(r) \leq N$ and $z(v) \geq 1$, we get $l \leq \log_{\lceil M/2 \rceil} N = \frac{\log N}{\log \lceil M/2 \rceil}$. □

COROLLARY 14. *If $M \geq 3$ then there is a degree $M/2$ solution of depth $O(\log N)$.*

COROLLARY 15. *If $M = \Omega(N^\varepsilon)$ then there is a degree $M/2$ solution of depth $O(1/\varepsilon)$.*

DEFINITION 16 (UNFOLDED TREE). *There is a vertex v_π in the* unfolded tree of depth *h for any* simple *path π of length at most h starting from a source. These will include all the paths of length zero, which correspond to the set of sources in G. There is an edge from v_π to $v_{\pi'}$ if and only if $\pi' = \langle \pi, e \rangle$ is the concatenation of π and some edge e. We say a vertex v_π in G' is a* copy *of vertex u in G if π ends at u.*

If T^* has depth h, then it can be mapped to a solution on the unfolded tree of depth h such that for every vertex v, at most one copy of v is included in this solution. In the reverse direction, consider any solution on the unfolded tree of depth h such that for every vertex v, at most one copy of v is picked in the solution. Then this solution can be mapped to a solution for the original instance. Hence, we focus on solving the instance corresponding to this unfolded tree, say G'. Note that the number of nodes in G', denoted by N', is $O(N^{h+1})$.

Now we are looking for a solution of minimum outdegree $M' = \lfloor M/2 \rfloor$ in G'. We write the LP as follows. Add a virtual node r' that has edges to all the sources in G'. There is a nonnegative variable x_e for each edge e. We stipulate that for any edge from r' to a source, $x_e = 1$. We denote by R the set of sources in G'. Define $p(v)$ to be the parent edge of v, for any vertex v of G'. For any vertex v in G' and a vertex v' of G, let $N^v(v')$ denote the copies of v' that are descendants of v in G'.

The complete linear program, AR-LP, is shown below. Let $v \in V(G')$ correspond to a path π in G. Then, $x_{p(v)}$ denotes the extent to which the path π appears in the solution.

(AR-LP)

$$\sum_{e \in \delta^+(v)} x_e \geq M \cdot x_{p(v)} \qquad \forall v \in V(G'), v \neq r' \quad (9)$$

$$\sum_{u \in N^v(v')} x_{p(u)} \leq x_{p(v)} \qquad \forall v \in V(G'), v' \in V(G) \quad (10)$$

$$x_{(r',v)} = 1 \qquad \forall v \in R \quad (11)$$

$$x_e \geq 0 \qquad \forall e \in E(G). \quad (12)$$

We first show that this is a valid relaxation for the problem. Consider an M-pyramid of depth h in the original graph. For any vertex v included in the pyramid, the path from source to v can be mapped to a path in the unfolded tree. Take the union of all such paths in the unfolded tree and set $x_e = 1$ for all edges in this union of paths. For all remaining edges e, $x_e = 0$. Note that for any original vertex v', at most one copy of v' is included in this solution. It is easy to verify that this solution satisfies all the constraints of AR-LP. Constraint (9) corresponds to the fact that every non-terminal v in the solution, the outdegree of v is at least M. Constraint 10 follows from the fact that at most one copy of a vertex v is included in the solution, and $x_e = 1$ for all edges e along some path from a source to v.

TREEROUND.

The rounding algorithm, TREEROUND, is as follows. We maintain a queue of vertices; these are vertices included in the solution produced so far. Initially, the queue consists of all sources in the graph. Draw vertices from this queue until the queue is empty, and perform the following steps for each vertex drawn. Suppose v is the vertex drawn from the queue. Recall that $p(v)$ denotes the parent edge of v. Pick a random subset of the outgoing edges from v as follows: pick outgoing edge e with probability $x_e/x_{p(v)}$ (where the choices are made independently for each edge). Notice that because of (10) and (12) these are valid probability values. We say that the current vertex claims every outgoing edge that is picked. (Note that this vertex will eventually receive a subset of the edges it claims.) Now the endpoints of the picked edges are placed on the queue if they are not sinks of G'. Roughly speaking, the rounding algorithm follows a *chain reaction*: when a node is activated, it can trigger its children. The LP variables model these reaction chains.

By (9) each vertex picked claims a subset of vertices of expected size at least M. By Chernoff bounds, with high probability, for every such vertex, the actual number of claimed neighbors is $\Omega(M)$ (assuming $M = \Omega(\log N')$, where N' is the number of vertices in G'). However, several copies of a vertex could be included in the solution. Using union bound and Lemma 17 we prove that with high probability, i.e., $1 - 1/\text{poly}(N)$, no more than $O(\log N \log N')$ copies of any vertex are included in the solution produced. We first establish this fact and then show how to produce a feasible solution by eliminating multiple copies of the same vertex.

LEMMA 17. *For any original vertex j of G, with probability $1 - \frac{1}{poly(N)}$, the number of copies of j included in the solution produced is $O(\log N' \log N)$.*

PROOF. In this discussion, c is an appropriate constant parameter. First, we make a transformation to eliminate small x_e values. For any $x_e < 1/N'^{c+1}$, we set its value to 0. Note that the probability that vertex i claims vertex j is exactly x_e if $e = (i, j)$. Consider the rounding procedure applied to the old x_e values versus the rounding procedure applied to the new x_e values. The total variational distance between the distribution of outcomes of the rounding procedure on the two different sets of x_e values is at most $2/N'^c$. Henceforth, we analyze the rounding procedure applied to the transformed x_e values; here any non-zero x_e has value at least $1/N'^{c+1}$.

For a subtree T rooted at v, let $X(T)$ be the random variable denoting the number of realized edges in T to copies of j given that the parent edge of T is chosen in the rounding process, i.e., $X(T)$ is the number of times that copies of j are included in the solution produced within T. Let $S(T) = \mathbf{E}[e^{\alpha X(T)}]$. Let $p(T) = x_{p(v)}$; note that $p(v)$ is the parent edge of tree T. Let $f(T)$ denote the total usage of copies of a vertex j in T, i.e., $f(T) = \sum_{u \in N^v(j)} x_{p(u)}$.

Algorithm MAXMINDEGREE ARBORESCENCE SOLVER
Input: *The digraph G*
Output: *The pyramid T*

1. If $M \leq M^*$, produce a solution with outdegree one. This can be simply done using a matching algorithm.
2. Otherwise, build the unfolded tree G' of depth $H = \log N / \log(M^*/2)$.
3. Solve AR-LP for G'.
4. Round it using TREEALG: select the sources; while there is a non-sink node v selected but not yet handled, pick each edge e of v independently at random with probability $x_e/x_{p(v)}$.
5. By having an assignment of $1/\gamma = 1/O(\log N \log N')$ for each edge of the solution of the previous step, build a fractional solution to an LP for $\frac{M}{\gamma}$-matching, and obtain an integral solution thereafter. Take the union of all edges of the integral solution that are reachable from sources.

Figure 3: The algorithm for MaxMinDegree Arborescence problem.

Note that the LP enforces the constraint (10) that $f(T) \leq p(T)$. We choose α such that $e^{\alpha} = 1 + 1/(4(c+1)\log N')$. The following claim is proved in the appendix.

CLAIM 18. *For any subtree T such that $f(T) > 0$,*

$$S(T) \leq 1 + \frac{1}{4(c+1)\log N'} \frac{f(T)}{p(T)} \left(1 + \frac{\log\left(N'^{c+1} f(T)\right)}{(c+1)\log N'}\right).$$

Let X be the total number of copies of j included in the solution produced. We have proved that $\mathbf{E}[e^{\alpha X}] \leq 2$, for $\alpha = \Theta(1/((c+1)\log N'))$, which implies that $\Pr[X > c(c+1)\log N' \log N] \leq 2/N^c$. □

At the end, there might be vertices v of G more than one copies of which are used. We remedy this issue by building a fractional solution to a natural linear program of k-matching and finding an integral solution.[4] The entire algorithm is summarized in Figure 3. We pick the parameter $M^* = \Omega(\log^3 N / \log\log N)$ according to the desired approximation ratio. If $M < M^*$, we can find a solution of value one which is an M^*-approximation. Otherwise, we show that the solution produced via the algorithm is good.

LEMMA 19. *If $M \geq M^*$ then the above algorithm runs in time $O(N^H)$ and with high probability produces an M''-pyramid with $M'' = \Omega(M \log\log N / \log^3 N)$ for G.*

PROOF. We set $H = \log N / \log(M^*/2)$ as the depth of the unfolded tree. Since $M \geq M^*$, Lemma 13 guarantees that there exists an M'-pyramid in the unfolded tree of depth H for $M' = M/2$. The algorithm clearly runs in time polynomial in $N' = N^{O(H)}$. We also know that $H = O(\log N / \log\log N)$ from the choice of M^*.

We next show each vertex is good: if selected in the process, it has *enough* outgoing edges. By the first set of constraints (9), the expectation of the number of outgoing edges of such a vertex is at least M'. A simple application of Chernoff bounds and union bound shows that since $M = \Omega(\log N')$, with high probability, say $1 - 1/\text{poly}(N')$, all nontrivial outdegrees are $\Omega(M)$.

There are N vertices in G whose indegree we want to bound. From union bound and Lemma (17), we get that with probability $1 - 1/\text{poly}(N)$, no vertex of G has indegree more than $\gamma = O(\log N \log N') = O(H \log^2 N) = O(\log^3 N / \log\log N)$.

Having shown that all indegrees are small, we build a fractional solution to an instance of MaxMin Allocation where all the item values are 0 and 1; it is indeed a bipartite M''-matching LP, for $M'' = \Omega(M/\gamma)$. We just need to divide all the variables by a factor no more than the indegree upper bound, γ. We have a fractional solution of value M'', and the polytope is integral. Thus we can find an integral solution of the same value, which implies an $M/O(\log^3 N / \log\log n)$-pyramid for G. □

The following is an immediate corollary. We set $M^* = \Theta(N^{\varepsilon})$ to get the polynomial running time or pick $M^* = \Theta(\log^3 N / \log\log N)$ for the polylogarithmic approximation ratio.

THEOREM 20. *For any integer $t > 1$, we can get a $\max\{\Omega(\log^3 N / \log\log N), 2N^{1/t}\}$-approximation in $N^{O(t)}$ time. In particular, we get an $\Omega(\log^3 N / \log\log N)$-approximation in $N^{O(\log n)}$ time where $N = O(n+m)$.*

5. CONCLUSION

We briefly describe extensions to the results in this extended abstract that appear in the full version of the paper [5]. In partial progress towards the general case, we can extend our results to more general instances, i.e., NoInfCycle and InfDegreeTwo. The class of instances NoInfCycle are those whose graphic representation does not contain a cycle consisting merely of infinity edges. Here, we have to cope with the subtleties corresponding to the handling of the infinity components. Along the way, we prove a result about the existence of a perfect matching in a probabilistically pruned graph. We show that given a fractional matching LP solution for a bipartite graph in which vertices on one side can have positive *deficiency*, deleting vertices with probabilities proportional to their deficiencies leaves us with a graph admitting a perfect matching. Another nuance is dealing with the event of assigning items in an infinity component to players outside the component. Players in the infinity component may no longer be satisfiable with the remaining items from the component and this may trigger some of them to seek small items from outside. This is the main point of departure from our InfDegreeOne algorithm, since these effects may need to propagate along long chains. The bounded length of *reaction chains* is crucial in our InfDegreeOne algorithm. Thus to extend the results to InfDegreeTwo, we need to deal with this very issue. We finally show that InfDegreeTwo instances are similar to NoInfCycle. Were we able to eliminate the *infinity cycles* in some way, we could derive the same result for the general case of MaxMin Allocation, matching the results of [7].

The most important question left open by our work is what we can do for DegreeThree—the general case. The recent results of Chakrabarty et al. [7] almost settles this question. It would still be interesting to see if our ideas can be extended to solve the general case.

Another interesting question is whether these ideas can be carried over to the MinMax problem. There, the gap between upper

[4] In the k-matching problem, we are given a bipartite graph and each vertex in the first partition is to receive k exclusive vertices from the second partition. To see that the polytope is indeed integral, note that one can build a maximum flow instance out of it by adding a source vertex which is connected to all vertices of the first partition, and a sink which has edges from all the second partition vertices. Then, we can put capacity k on the edges going out of the source and capacity one on all the other edges. Feasibility of the original k-matching instances leads to existence of a flow that saturates all the edges out of source.

and lower bounds is considerably smaller, yet improving on the long-standing factor 2 upper bound would be very exciting. It will also be interesting to gain a better understanding of the Sherali-Adams lift-and-project LP for MaxMin Allocation. In particular, finding a tight integrality gap is a nice direction to pursue.

Getting better approximation ratios and/or running times is another worthwhile direction. As the MaxMinDegree Arborescence problem seems to capture the essence of the MaxMin Allocation problem, a better understanding of it could lead to stronger hardness results or better approximation ratios.

6. REFERENCES

[1] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[2] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. In *11th APPROX*, pages 10–20, 2008.

[3] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *39th STOC*, pages 114–121, 2007.

[4] N. Bansal and M. Sviridenko. The santa claus problem. In *38th STOC*, pages 31–40, 2006.

[5] M. Bateni, M. Charikar, and V. Guruswami. New approximation algorithms for degree lower-bounded arborescences and max-min allocation. Technical Report TR-848-09, Computer Science Department, Princeton University, March 2009.

[6] I. Bezáková and V. Dani. Allocating indivisible goods. *SIGecom Exchange*, 5(3):11–18, 2005.

[7] D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. *CoRR*, abs/0901.0205, 2009.

[8] T. Ebenlendr, M. Krčal, and J. Sgall. Graph balancing: A special case of scheduling unrelated parallel machines. In *18th SODA*, pages 483–490, 2008.

[9] U. Feige. On allocations that maximize fairness. In *18th SODA*, pages 287–293, 2008.

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

[11] S. Khot and A. K. Ponnuswami. Approximation algorithms for the max-min allocation problem. In *10th APPROX*, pages 204–217, 2007.

[12] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

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

APPENDIX

PROOF OF CLAIM 18. The proof goes by induction on the height of the subtree T. For convenience, we define the base case to be a tree of height 0, where the parent edge of the tree is an edge e such that $x_e > 0$. For this tree $f(T) = p(T) = x_e$, $X(T)$ is always 1 and

$$\mathbf{E}\left[e^{\alpha X(T)}\right] = e^{\alpha} = 1 + \frac{1}{4(c+1)\log N'}.$$

The base case thus follows from $f(T) \geq 1/N'^{c+1}$.

Consider a subtree T consisting of a root connected to subtrees $T_1, \ldots, T_k$, via edges $e_1, \ldots, e_k$ respectively. Let e be the parent edge of T. Suppose the inductive hypothesis holds for each subtree T_i, i.e.,

$$S(T_i) \leq 1 + \frac{1}{4(c+1)\log N'} \frac{f(T_i)}{p(T_i)} \left(1 + \frac{\log\left(N'^{c+1} f(T_i)\right)}{(c+1)\log N'}\right).$$

Then, as the $X(T_i)$'s and the selections of e_i's conditioned on e being selected are independent, we get

$$\begin{aligned} S(T) &= \prod_i \left[\left(1 - \frac{p(T_i)}{p(T)}\right) \cdot 1 + \frac{p(T_i)}{p(T)} S(T_i)\right] \\ &= \prod_i \left[1 + \frac{p(T_i)}{p(T)} \left(S(T_i) - 1\right)\right] \\ &\leq \prod_i (1 + \delta_i), \end{aligned} \tag{13}$$

with δ_i defined as

$$\begin{aligned} \delta_i &= \frac{1}{4(c+1)\log N'} \frac{f(T_i)}{p(T)} \left(1 + \frac{\log\left(N'^{c+1} f(T_i)\right)}{(c+1)\log N'}\right) \quad (14) \\ &\leq \frac{1}{2(c+1)\log N'} \frac{f(T_i)}{p(T)}, \quad (15) \end{aligned}$$

where the last inequality is due to $f(T_i) \leq 1$. Thus,

$$\begin{aligned} \sum_i \delta_i &\leq \frac{1}{2(c+1)\log N'} \frac{f(T)}{p(T)} \\ &\leq \frac{1}{2(c+1)\log N'} \\ &\leq \frac{1}{2}. \end{aligned}$$

Hence,

$$\prod_i (1 + \delta_i) \leq \prod_i e^{\delta_i} = e^{\sum_i \delta_i} \leq 2.$$

Now, by Inequality (13),

$$\begin{aligned} S(T) &\leq 1 + \sum_i \delta_i + \frac{1}{2} \left(\sum_{i_1 \neq i_2} \delta_{i_1} \delta_{i_2}\right) \prod_i (1 + \delta_i) \\ &\leq 1 + \sum_i \delta_i + \sum_i \delta_i \sum_{i' \neq i} \delta_{i'}. \end{aligned}$$

Note that, by Equation (14),

$$\begin{aligned} 1 + \sum_i \delta_i = 1 &+ \frac{1}{4(c+1)\log N'} \frac{f(T)}{p(T)} \\ &+ \frac{1}{4(c+1)^2 \log^2(N') p(T)} \sum_i f(T_i) \log(N'^{c+1} f(T_i)), \end{aligned}$$

and by Inequality (15),

$$\begin{aligned} \sum_{i' \neq i} \delta_{i'} &\leq \frac{1}{2(c+1)\log N'} \sum_{i' \neq i} \frac{f(T_{i'})}{p(T)} \\ &= \frac{1}{2(c+1)\log N'} \frac{f(T) - f(T_i)}{p(T)} \\ &\leq \frac{1}{2(c+1)\log N'} \int_{f(T_i)}^{f(T)} \frac{1}{x} dx \\ &= \frac{1}{2(c+1)\log N'} \log \frac{f(T)}{f(T_i)}. \end{aligned}$$

Hence, by Inequality 15,

$$\delta_i \sum_{i' \neq i} \delta_{i'} \leq \frac{1}{4(c+1)^2 \log^2(N') p(T)} f(T_i) \log \frac{f(T)}{f(T_i)},$$

and we get

$$\begin{aligned}
S(T) &\leq 1+\sum_i \delta_i + \sum_i \delta_i \sum_{i'\neq i} \delta_{i'} \\
&\leq 1+\frac{1}{4(c+1)\log N'}\frac{f(T)}{p(T)} \\
&\quad +\frac{1}{4(c+1)^2\log^2(N')p(T)}\sum_i f(T_i) \\
&\qquad \cdot\left(\log(N'^{c+1}f(T_i))+\log\frac{f(T)}{f(T_i)}\right) \\
&= 1+\frac{1}{4(c+1)\log N'}\frac{f(T)}{p(T)} \\
&\quad +\frac{1}{4(c+1)^2\log^2(N')p(T)}\sum_i f(T_i)\log(N'^{c+1}f(T)) \\
&\leq 1+\frac{1}{4(c+1)\log N'}\frac{f(T)}{p(T)}\left(1+\frac{\log(N'^{c+1}f(T))}{(c+1)\log N'}\right). \quad \square
\end{aligned}$$

THEOREM 21. *An M-LP solution of value M can be translated in polynomial time to a Config-LP solution of value $M/2$.*

PROOF. The proof is constructive. First revalue any item of value $\geq M/2$ to M. Consequently, decrease the value of z_i by $\sum_{j:p_{ij}\geq M/2} x_{ij}$, and then decrease the x_{ij} to $\min(z_i, x_{ij})$ if $p_{ij} < M/2$. It can be easily verified that the new values show a valid M-LP solution and each small item has value at most $p^{\max} \leq M/2$.

We leave the assignment of big items unaffected. Take any player i and its corresponding small items. Ideally, they should build z_i units of small configurations of value M. We show that, compromising on the value of configurations to be $M - p^{\max}$, we can achieve this task. We restate the set of constraints we need to work on, and remove unnecessary indices.

$$\sum_j p_j x_j \geq zM \tag{16}$$

$$x_j \leq z \qquad \forall j \tag{17}$$

$$x_j \geq 0 \qquad \forall j. \tag{18}$$

The algorithm proceeds by creating bundles and reducing x_j's and z accordingly. At any step, it maintains that the current values of x_j and z are feasible in the above set of equations 16-18; if z^* and x_j^* are the initial values, we have made at least $z^* - z$ units of bundles where each item j has been used for at most $x_j^* - x_j$ units. At each step, we rename items so that x_j's are nonincreasing. We know from (16) and (17) that the sum of the values of the remaining items is at least M; since, otherwise, $\sum_j p_j x_j \leq \sum_j p_j \sum_{j'} x_{j'} < zM$ leading to a contradiction. Let q be the smallest index such that $\sum_{j\leq q} p_j \geq M$. We will do one of the following until $z = 0$, in which case we are done.

- If $x_q \neq x_1$, make a bundle out of the first $q-1$ items. Use this bundle for $\delta = \min\{x_{q-1}, x_1 - x_q\}$ units. Decrease concerning z and the involved x_j variables by δ. Notice that $\delta > 0$, since $x_{q-1} \geq x_q > 0$ and $x_1 \neq x_q$. The value of this bundle is at least $M - p_{\max}$ and at most M. So the left-hand side of Inequality (16) is reduced by at most δM which is the decrease in the right-hand side. By the choice of δ, the maximum x_j after this operation is at most $x_1 - \delta \leq z - \delta$. Hence, Inequality (17) is still valid after this operation.
- If $x_q = x_1$, we say we have a *plateau*. Let q' be the largest index where $x_1 = x_{q'}$. In this case, $q' \geq q$ and we cannot simply make a bundle out of the first $q-1$ items, because then the maximum x_j would not change, whereas this change may be necessary to keep (17) satisfied. The trick is to decrease x_j for all variables equal to x_1 simultaneously.

  To this end, we build a *balanced* collection of bundles which in total contains each item of our concern (i.e., all items j such that $x_j = x_1$) exactly r times. The size of each bundle is at most M. We then decrease the corresponding x_j's by $\delta = x_1 - x_{q'+1}$; i.e., we use each configuration for δ/r units. We let $z = \frac{1}{M}\sum_j p_j x_j$ after this operation. It is easy to see that we make at least $\frac{1}{M}\sum_{j\leq q'} p_j \delta \geq z^{\text{old}} - z^{\text{new}}$ units of configuration in this step. The solution is thus feasible since $z \geq \frac{1}{M}\sum_{j\leq q'} p_j x_1 \geq x_1$ and x_1 is the maximum x_j value.

  Building the balanced collection of bundles is done as follows. Take the group of items attaining the same maximum value. We know that their sum of values is at least M (by the choice of q). Start from the first one and include items until we make a bundle of value at least $M - p_{\max}$. Then, start from the next element and do the same thing (possibly wrapping around), and so on. We repeat this bundle-making process until a bundle is repeated. This has to happen because there would be at most m different bundles formed in such a process. Then, we can take the collection of bundles from the first occurrence of this bundle up until (but not including) its second occurrence. Clearly, each item appears the same number of times in these bundles. An example is shown in Figure 4.

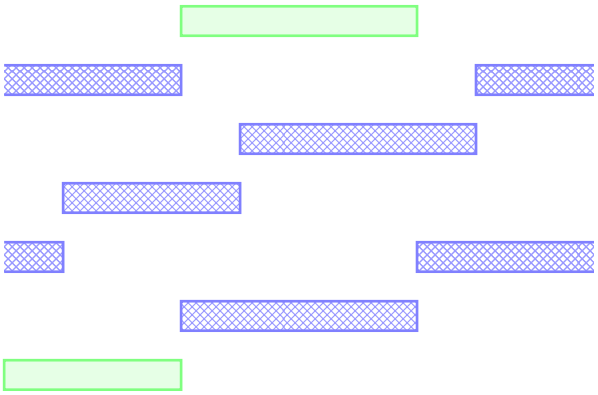

Figure 4: Making bundles in case of a plateau: items are put in a line, and we start from the left and find maximal bundles of value at most M. Each row in the picture shows one bundle. Note how each bundle starts after the previous one, and how we wrap around at the end. Plain (green) bundles (first and last ones) are discarded. In this example, we end up with 5 bundles and each item is covered exactly twice.

Performing this two-stage procedure, we can form a Config-LP solution from any M-LP solution, provided that we allow configuration values to be as low as $M - p_{\max}$. Finiteness follows from the fact that at each step, either some variables go zero, or the number of variables attaining the maximum goes up. $\square$

So, there will be at most $O(m)$ steps and each step adds at most m bundles. Thus, the total number of bundles produced for each player is bounded by $O(m^2)$ which is a polynomial.