On the Approximability of Budgeted Allocations and Improved Lower Bounds for Submodular Welfare Maximization and GAP [*]

Deeparnab Chakrabarty
Georgia Tech
deepc@cc.gatech.edu

Gagan Goel
Georgia Tech
gagang@cc.gatech.edu

Abstract

In this paper we consider the following maximum budgeted allocation*(MBA) problem: Given a set of m indivisible items and n agents; each agent i willing to pay b_{ij} on item j and with a maximum budget of B_i, the goal is to allocate items to agents to maximize revenue.*

The problem naturally arises as auctioneer revenue maximization in budget-constrained auctions and as winner determination problem in combinatorial auctions when utilities of agents are budgeted-additive. Our main results are:

- *We give a $3/4$-approximation algorithm for MBA improving upon the previous best of $\simeq 0.632$[2, 10]. Our techniques are based on a natural LP relaxation of MBA and our factor is optimal in the sense that it matches the integrality gap of the LP.*

- *We prove it is NP-hard to approximate MBA to any factor better than $15/16$, previously only NP-hardness was known [21, 17]. Our result also implies NP-hardness of approximating maximum submodular welfare with* demand oracle *to a factor better than $15/16$, improving upon the best known hardness of $275/276$[10].*

- *Our hardness techniques can be modified to prove that it is NP-hard to approximate the* Generalized Assignment Problem *(GAP) to any factor better than $10/11$. This improves upon the $422/423$ hardness of [7, 9].*

We use iterative rounding *on a natural LP relaxation of MBA to obtain the $3/4$-approximation. We also give a $(3/4-\epsilon)$-factor algorithm based on the primal-dual schema which runs in $\tilde{O}(nm)$ time, for any constant $\epsilon > 0$.*

1. Introduction

Resource allocation problems of distributing a fixed supply of resources to multiple agents in an "optimal" manner are ubiquitous in computer science and economics. In this paper we consider the following *maximum budgeted allocation* (MBA) problem: Given a set of m indivisible items and n agents; each agent i willing to pay b_{ij} on item j and with a maximum budget of B_i, the goal is to allocate items to agents to maximize revenue.

The problem naturally arises as a revenue maximization problem for the auctioneer in an auction with budgeted agents. Examples of such auctions (see, for example [4]) include those used for the privatization of public assets in western Europe, or those for the distribution of radio spectra in the US, where the magnitude of the transactions involved put financial or liquidity constraints on bidders. With the growth of the Internet, budget-constrained auctions have gained increasing relevance. Firstly, e-auctions held on the web (on e-Bay, for instance) cater to the long-tail of users who are inherently budget-constrained. Secondly, sponsored search auctions hosted by search engines (Google, Yahoo!, MSN and the like) where advertisers bid on keywords include budget specification as a feature. A common (and natural) assumption in keyword auctions that is typically made is that bids of advertisers are much smaller than the budgets. However, with the extension of the sponsored search medium from the web onto the more classical media, such as radio and television, where this assumption is not as reasonable, the general budget-constrained auctions need to be addressed.

MBA is known to be NP-hard - even in the case of two bidders it is not hard to see that MBA encodes PARTITION. In this paper we study the approximability of MBA and improve upon the best known approximation and hardness of approximation factors. Moreover, we use our hardness reductions to get better hardness results for other allocation problems like submodular welfare maximization(SWM), generalized assignment problem (GAP) and maximum spanning star-forest (MSSF).

[*]Research partially supported by NSF Grant CCF-0728640.

1.1 Maximum Budgeted Allocation

We start with the formal problem definition.

Definition 1 *Let Q and A be a set of m indivisible items and n agents respectively, with agent i willing to pay b_{ij} for item j. Each agent i has a budget constraint B_i and on receiving a set $S \subseteq Q$ of items, pays $\min(B_i, \sum_{j\in S} b_{ij})$. An* allocation *$\Gamma : A \to 2^Q$ is the partition of the set Q into disjoint sets $\Gamma(1), \cdots, \Gamma(n)$. The* maximum budgeted allocation *problem, or simply MBA, is to find the allocation which maximizes the total revenue, that is, $\sum_{i\in A} \min(B_i, \sum_{j\in\Gamma(i)} b_{ij})$.*

Note that we can assume without loss of generality that $b_{ij} \leq B_i$, $\forall i \in A, j \in Q$. This is because if bids are larger than budget, decreasing it to the budget does not change the value of any allocation. Sometimes, motivated by the application, one can add the constraint that $b_{ij} \leq \beta \cdot B_i$ for all $i \in A$ and $j \in Q$, for some $\beta \leq 1$. We call such an instance β-MBA.

Previous and Related Work: As noted above, MBA is NP-hard and this observation was made concurrently by many authors ([12, 21, 2, 17]). The first approximation algorithm for the problem was given by Garg, Kumar and Pandit[12] who gave a $2/(1+\sqrt{5})(\simeq 0.618)$ factor approximation. Andelman and Mansour[2] improved the factor to $(1-1/e)(\simeq 0.632)$. For the special case when budgets of all bidders were equal, [2] improved the factor to 0.717. We refer to the thesis of Andelman[1] for an exposition.

Remark 1.1 *Very recently, and independent of our work, Azar et.al. [3] obtained a $2/3$-factor for the MBA problem, and subsequently Srinivasan [23] extended their algorithm to obtain a $3/4$-factor for MBA. Thus along with our two algorithms, there are three different $3/4$-factor approximation algorithms for MBA.*

In the setting of sponsored search auctions, MBA, or rather β-MBA with $\beta \to 0$, has been studied mainly in an online context. Mehta et.al.[19] and later, Buchbinder et.al.[6] gave $(1-1/e)$-competitive algorithms when the assumption of bids being small to budget is made. The dependence of the factor on β is not quite clear from either of the works. Moreover, as per our knowledge, nothing better was known the approximability of the *offline* β-MBA than what was suggested by algorithms for MBA.

Our results: We give two approximation algorithms for MBA. The first, based on iterative LP rounding, attains a factor of $3/4$. The algorithm is described in Section 2. The second algorithm, based on the primal-dual schema, is faster and attains a factor of $3/4-\epsilon$, for any $\epsilon > 0$. The running time of the algorithm is $\tilde{O}(\frac{nm}{\epsilon})$. We describe the algorithm in Section 3. Our algorithms can be extended suitably for β-MBA as well giving a $1-\beta/4$ factor approximation algorithm.

In Section 4, we show it is NP hard to approximate MBA to a factor better than $15/16$ via a gap-preserving reduction from MAX-3-LIN(2). Our hardness instances are *uniform* in the sense of Azar et.al.[*] [3] implying uniform MBA is as hard. Our hardness reductions extend to give a $(1-\beta/16)$ hardness for β-MBA as well. Interestingly, our reductions can be used to obtain better inapproximability results for other problems: SWM ($15/16$ hardness even with demand queries), GAP ($10/11$ hardness) and MSSF($10/11$ and $13/14$ for the edge and node weighted versions), which we elaborate below.

1.2 Relations to other allocation problems

Submodular Welfare Maximization (SWM): As in the definition of MBA, let Q be a set of m indivisible items and A be a set of n agents. For agent i, let $u_i : 2^Q \to \mathbb{R}_+$ be a utility function where for a subset of items $S \subseteq Q$, $u_i(S)$ denote the utility obtained by agent i when S is allocated to it. Given an allocation of items to agents, the total *social welfare* is the sum of utilities of the agents. The *welfare maximization* problem is to find an allocation of maximum social welfare.

Welfare maximization problems have been extensively studied (see, for example, [5]) in the past few years with various assumptions made on the utility functions. One important set of utility functions are *monotone submodular utility functions*. A utility function u_i is monotone if $u_i(S) \leq u_i(T)$ whenever $S \subseteq T$. A utility function u_i is submodular if for any two subsets S, T of items, $u_i(S\cup T)+u_i(S\cap T) \leq u_i(S)+u_i(T)$. The welfare maximization problem when all the utility functions are submodular is called the submodular welfare maximization problem or simply SWM. In many cases, explicit representation of these functions becomes an issue. Thus an access to an oracle is generally assumed (see [5, 24, 10]). One such oracle is the *demand oracle* which for any agent i and prices $p_1, p_2, \cdots, p_m$ for all items in Q, returns a subset $S \subseteq Q$ which maximizes $(u_i(S) - \sum_{j\in S} p_j)$. Given an access to a *demand oracle*, Feige and Vondrák [10] gave a $(1-1/e+\rho)$-approximation for SWM with $\rho \sim 0.0001$ and showed that it is NP-hard to approximate SWM to better than $275/276$.[†]

MBA is a special case of SWM. This follows from the observation that the utility function

[*]An MBA instance is uniform if $b_{ij} = b_j$ whenever $b_{ij} > 0$.

[†]We remark that SWM with a different oracle, the *value oracle*, has recently been resolved. There was a $(1-1/e)$ hardness given by Khot et.al.[15] and recently Vondrak[24] gave a matching polynomial time algorithm.

$u_i(S) = \min(B_i, \sum_{j\in S} b_{ij})$ when B_i, b_{ij}'s are fixed is a submodular function. In Section 4, we show that in the hardness instances of MBA, the demand oracle can be simulated in poly-time and therefore the $15/16$ hardness of approximation for MBA implies a $15/16$-hardness of approximation for SWM with the *demand oracle*.

Generalized Assignment Problem (GAP): GAP is a problem related to MBA: Every item j, along with the bid (profit) b_{ij} for agent (bin) i, also has an inherent size s_{ij}. Instead of a budget constraint, each agent (bin) has a capacity constraint C_i which defines feasible sets: A set S is feasible for (bin) i if $\sum_{j\in S} s_{ij} \le C_i$. The goal is to find a revenue (profit) maximizing feasible assignment. The main difference between GAP and MBA is that in GAP we are *not allowed* to violate capacity constraints, while in MBA the budget constraint only caps the revenue. As was noted by Chekuri and Khanna[7], a $1/2$ approximation algorithm was implicit in the work of Shmoys and Tardos[22]. The factor was improved by Fleischer et.al.[11] to $1 - 1/e$. In the same paper where they give the best known algorithm for SWM, Feige and Vondrák[10] also give a $(1-1/e+\rho')$ algorithm for GAP ($\rho' \le 10^{-5}$). The best known hardness for GAP was $1-\epsilon$, for some small ϵ which was given by Chekuri and Khanna [7] via a reduction from maximum 3D-matching. Improved hardness results for maximum 3D matching by Chlebik and Chlebikova[9], imply a $422/423$ hardness for GAP.

Although MBA and GAP are in some sense incomparable problems, we can use our hardness techniques to get a $10/11$ factor hardness of approximation for GAP.

Maximum Spanning Star-Forest Problem (MSSF): Given an undirected unweighted graph G, the MSSF problem is to find a forest with as many edges such that each tree in the forest is a star — all but at most one vertex of the tree are leaves. The edge-weighted MSSF is the natural generalization with weights on edges. The node-weighted MSSF has weights on vertices and the weight of a star is the weight on the leaves. If the star is just an edge, then the weight of the star is the maximum of the weights of the end points.

The unweighted and edge-weighted MSSF was introduced by Nguyen et.al [20] who gave a $3/5$ and $1/2$-approximation respectively for the problems. They also showed APX hardness of the unweighted version. Chen et.al. [8] improved the factor of unweighted MSSF to 0.71 and introduced node-weighted MSSF giving a 0.64 factor algorithm for it. They also give a $31/32$ and $19/20$ hardness for the node-weighted and edge-weighted MSSF problems.

Although, at the face of it, MSSF does not seem to have a relation with MBA, once again our hardness techniques can be used to improve the hardness of node-weighted and edge-weighted MSSF to $13/14$ and $10/11$, respectively.

1.3 The LP Relaxation for MBA

One way to formulate MBA as an integer program is the following:

$$\max\{\sum_{i\in A}\pi_i : \pi_i = \min(B_i, \sum_{j\in Q} b_{ij}x_{ij}),\ \forall i;$$

$$\sum_{i\in A} x_{ij} \le 1, \forall j;\ \ x_{ij}\in\{0,1\}\ \}$$

Relaxing the integrality constraints to non-negativity constraints gives an LP relaxation for the problem. We work with the following equivalent LP relaxation of the problem. The equivalence follows by noting that in there exists an optimal fractional solution, $B_i \ge \sum_{j\in Q} b_{ij}x_{ij}$. This was noted by Andelman and Mansour[2] and a similar relaxation was used by Garg et.al. [12].

$$\max\{\sum_{i\in A, j\in Q} b_{ij}x_{ij} :\ \ \forall i \in A,\ \sum_{j\in Q} b_{ij}x_{ij} \le B_i;\quad (1)$$
$$\forall j\in Q,\ \sum_{i\in A} x_{ij} \le 1;\ \ \forall i\in A, j\in Q,\ x_{ij}\ge 0\}$$

We remark that the assumption $b_{ij} \le B_i$ is crucial for this LP to be of any use. Without this assumption it is easy to construct examples having arbitrarily high integrality gaps. Consider the instance with one item, n agents each having a budget 1 but bidding n on the item. The LP has a solution of value n while the maximum welfare is obviously 1.

1.4 Technical Contributions

Apart from being an important problem in its own right, we believe the MBA problem is interesting as it allows us to enhance certain algorithmic ideas. As we mention above, one of our algorithms is an iterative rounding algorithm using LP(1). Recently, the technique of iterative rounding has achieved considerable success in designing approximation algorithms [14, 16]. However, these successes have been limited to minimization problems, and as per our knowledge, this work is the first iterative rounding based result for a natural maximization problem.

The iterative rounding schema can be broadly described in the following two step procedure: find an optimal solution satisfying some *desirable property* (for instance, some variable has value 1 or at least $1/2$ ([14])), then move to a *residual problem* after rounding some variables to 1 and iterate. The proof of approximation factor, say $\alpha > 1$, is shown by proving that the cost incurred by the algorithm in one iteration is at most α times the drop in the LP-value across the iteration.

In minimization problems, the constraints on the variables are normally covering constraints. Thus, a solution to the original problem is also a solution to the natural residual problem obtained after rounding up. The non-trivial part of most algorithms ([14, 16]) lies in proving the existence of a *desirable property.*

In this problem, the desirable property as we show later (this was observed by most previous works on MBA) is that $x_{ij} = 1$ for some agent i and item j. However, in the natural residual problem obtained after the item j is allocated to agent i, the LP might drop by as much as *twice* as the value obtained by the algorithm (giving only a $1/2$ factor). To overcome this difficulty, we show how to move to a *residual problem* in a non-trivial manner which allows us to control the LP-drop across iterations. We give the details in Section 2. We believe this technique of a clever definition of the residual problem might be an approach towards iterative rounding of other maximization problems like SWM and GAP. That said, it is also non-trivial for such problems even to get the desirable property of a solution; in fact it is unclear what a desirable property should be.

We also give a primal-dual algorithm for MBA in Section 3 which for any $\epsilon > 0$ gives a $(\frac{3}{4} - \epsilon)$ approximation factor. Primal-dual algorithms have been successful in obtaining approximation algorithms for minimization problems (whose duals are maximization). The typical schema is to start with an all-zeroes feasible dual solution and then raising the duals till some dual constraint goes tight which suggests which primal variable to pick. Since MBA is a maximization problem, the dual is a minimization and the all-zeroes solution is not feasible any more. We deviate from the schema by setting a *subset* of dual variables to zero and raising these variables. The remaining dual variables are set so as to maintain dual feasibility. Details can be found in Section 3.

2 An iterative rounding algorithm for MBA

Let $\mathcal{P}$ be a problem instance defined by the bids and budgets of every agent, that is $\mathcal{P} := (\{b_{ij}\}_{i,j}, \{B_i\}_i)$. With $\mathcal{P}$, we associate a bipartite graph $G(\mathcal{P}) = (A \cup Q, E)$, with $(i,j) \in E$ if i bids on j.

Let $x^*(\mathcal{P})$ be an extreme point solution to LP(1) for the problem instance $\mathcal{P}$. For brevity, we omit writing the dependence on $\mathcal{P}$ when the instance is clear from context. Let E^* be the support of the solution, that is $E^* := \{(i,j) \in E : x^*_{ij} > 0\}$. Note that these are the only *important* edges — one can discard all bids of agents i on item j when $(i,j) \notin E^*$. This does not change the LP optimum and a feasible integral solution in this instance is a feasible solution of the original instance. Call the set of neighbors of an agent i in $G[E^*]$ as $\Gamma(i)$. Call an agent *tight* if $\sum_j b_{ij}x^*_{ij} = B_i$.

The starting point of the algorithm is the following claim about the structure of the extreme point solution. Such an argument using polyhedral combinatorics, was first used in the machine scheduling paper of Lenstra, Shmoys and Tardos [18]. A similar claim can be found in the thesis of Andelman [1]. We omit the proof in the extended abstract.

Claim 2.1 *The graph, $G[E^*]$, induced by E^* can be assumed to be a forest. Moreover, except for at most one, all the leaves of a connected component are items. Also at most one agent in a connected component can be* non-tight.

Call an item a *leaf item*, if it is a leaf in $G[E^*]$. Also call an agent i a *leaf agent* if, except for at most one, all of his neighboring items in $E^*(\mathcal{P})$ are leaves. Note the above claim implies each connected component has at least one leaf item and one leaf agent: in any tree there are two leaves both of which cannot be agents, and there must be an agent with all but one of its neighbors leaves and thus leaf items. For the sake of understanding, we first discuss the following natural iterative algorithm which assigns the leaf items to their neighbors and then adjusts the budget and bids to get a new *residual problem.*

1/2-approx algorithm: Solve $LP(\mathcal{P})$ to get $x^*(\mathcal{P})$. Now pick a leaf agent i. Assign all the leaf items in $\Gamma(i)$ to i. Let j be the unique non-leaf item (if any) in $\Gamma(i)$. Form the new instance $\mathcal{P}'$ by removing $\Gamma(i) \setminus j$ and all incident edges from $\mathcal{P}'$. Let $b = \sum_{l \in \Gamma(i) \setminus j} b_{il}$, be the portion of budget spent by i. Now modify the budget of i and his bid on j in $\mathcal{P}'$ as follows: $B'_i := B_i - b$, and $b'_{ij} := \min(b_{ij}, B'_i)$. Its instructive to note the drop $(b_{ij} - b'_{ij})$ is at most b. (We use here the assumption bids are always smaller than budgets). Iterate on the instance $\mathcal{P}'$.

The above algorithm is a $1/2$-approximation algorithm. In every iteration, we show that the revenue generated by the items allocated is at least $1/2$ of the drop in the LP value $(LP(\mathcal{P}) - LP(\mathcal{P}'))$. Suppose in some iteration, i be the leaf agent chosen, and let j be the its non-leaf neighbor, and let the revenue generated by algorithm be b. Note that x^*, the solution to $\mathcal{P}$, restricted to the edges in $\mathcal{P}'$ is still a feasible solution. Thus the drop in the LP is: $b + (b_{ij} - b'_{ij})x_{ij}$. Since $(b_{ij} - b'_{ij})$ is atmost b, and x_{ij} at most 1, we get $LP(\mathcal{P}) - LP(\mathcal{P}') \leq 2b$.

To prove a better factor in the analysis, one way is to give a better bound on the drop, $(LP(\mathcal{P}) - LP(\mathcal{P}'))$. Unfortunately, the above analysis is almost tight and there exists an example where the LP drop in the first iteration is $\simeq$ twice the revenue generated by the algorithm in that iteration. Thus, for an improved analysis for this algorithm, one needs a better amortized analysis across different iterations rather than analyzing iteration-by-iteration. This seems non-trivial as we solve the LP again at each iteration and the solutions could be very different across iterations making it harder to analyze over iterations.

Instead, we modify the above algorithm by defining the *residual problem* $\mathcal{P}'$ in a non-trivial manner. After assigning leaf items to agent i, we do not decrease the budget by the amount assigned, but keep it a little "larger". Thus these agents *lie* about their budgets in the subsequent rounds, and we call these *lying* agents. Since the budget doesn't drop too much, the LP value of the residual problem doesn't drop much either. A possible trouble would arise when items are being assigned to lying agents since they do not pay as much as they have bid. This leads to a trade-off and we show by suitably modifying the residual problem one can get a $3/4$ approximation. We now elaborate.

Given a problem instance $\mathcal{P}_0 := \mathcal{P}$, the algorithm proceeds in stages producing newer instances at each stage. On going from $\mathcal{P}_i$ to $\mathcal{P}_{i+1}$, at least one item is allocated to some agent. Items are never de-allocated, thus the process ends in at most m stages. The *value* of an item is defined to be the payment made by the agent who gets it. That is, $value(j) = \min(b_{ij}, B_i - spent(i))$, where $spent(i)$ is the value of items allocated to i at the time j was being allocated. We will always ensure the condition that a lying agent i bids on at most one item j. We will call j the *false item* of i and the bid of i on j to be i's *false bid*. In the beginning no agent is lying.

We now describe the k-th iteration of the iterative algorithm which we call MBA-ITER (Algorithm 1).

Algorithm 1 k-th step of MBA-ITER

1. Solve $LP(\mathcal{P}_k)$. Remove all edges which are not in $E^*(\mathcal{P}_k)$. These edges will stay removed in all subsequent steps.

2. If there is a lying agent i with $x^*_{ij} = 1$ for his false item j, assign item j to him. In the next instance, $\mathcal{P}_{k+1}$, remove i and j. Proceed to $(k+1)$-th iteration.

3. If there is a non-lying agent i such that all the items in $\Gamma(i)$ are leaf items. Then allocate $\Gamma(i)$ to i. Remove i, $\Gamma(i)$ and all the incident edges to get the new instance $\mathcal{P}_{k+1}$ and proceed to $(k+1)$-th iteration step.

4. Pick a *tight* leaf agent i. Notice that i must have at least two items in $\Gamma(i)$, otherwise tightness would imply that the unique item is a leaf item and thus either step 2 or step 3 must have been performed. Moreover, exactly one item in $\Gamma(i)$ is not a leaf item, and let j be this unique non-leaf item. Allocate all the items in $\Gamma(i) \setminus j$ to i. In $\mathcal{P}_{k+1}$, remove $\Gamma(i) \setminus j$ and all incident edges. Also, modify the budget and bids of agent i. Note that agent i now bids *only* on item j as there are no other edges incident to i. Let the new bid of agent i on item j be

$$b'_{ij} := \max(0, \frac{4b_{ij}x^*_{ij} - B_i}{3x^*_{ij}})$$

Let the new budget of agent i be $B'_i := b'_{ij}$. Call i lying and j be his false item. Proceed to $(k+1)$-th iteration.

Claim 2.2 *In each step, at least one item is allocated and thus* MBA-ITER *terminates in m steps.*

Proof: We show that one of the three steps 2,3 or 4 is always performed and thus some item is always allocated. Consider any component. If a component has only one agent i, then all the items in $\Gamma(i)$ are leaf items. If $\Gamma(i)$ has at least two items, then the agent cannot be lying since the lying agent bids on only one item and Step 3 can be performed. If $\Gamma(i) = \{j\}$, then $x^*_{ij} = 1$ since otherwise x^*_{ij} could be increased giving a better solution. Thus Step 2 or 3 can always be performed depending on if i is lying or not. If the component has at least two agents, then it must have two leaf agents. This can be seen by rooting the tree at any item. At least one of them, say i, is tight by Claim 2.1. Thus Step 4 can be performed. □

Theorem 2.3 *Given a problem instance $\mathcal{P}$, the allocation obtained by algorithm* MBA-ITER *attains value at least $\frac{3}{4} \cdot LP(\mathcal{P})$.*

Proof: Let $\Delta_k := LP(\mathcal{P}_k) - LP(\mathcal{P}_{k+1})$ denote the drop in the optimum across the k-th iteration. Denote the set of items allocated at step k as Q_k. Note that the total value of the algorithm is $\sum_{j \in Q} value(j) = \sum_k (\sum_{j \in Q_k} value(j))$. Also, the LP optimum of the original solution is $LP(\mathcal{P}) = \sum_k \Delta_k$ since after the last item is allocated the LP value becomes 0. The following lemma proves the theorem. □

Lemma 2.4 *In every stage k, $value(Q_k) := \sum_{j \in Q_k} value(j) \geq \frac{3}{4}\Delta_k$.*

Proof: Items are assigned in either Step 2,3 or 4. Let us analyze Step 2 first. Let i be the lying agent obtaining his false item j. Since $x^*_{ij} = 1$ and lying agents bid on only one item, the remaining solution (keeping the same x^* on all remaining edges) is a valid solution for the LP in $\mathcal{P}_{k+1}$. Thus

$$LP(\mathcal{P}_k) - LP(\mathcal{P}_{k+1}) \leq b',$$

where b' is the false bid of lying agent i on item j. Let b be the bid of agent i on item j, before it was made lying.

Then, from Step 4 we know that $b' := \frac{4bx-B}{3x}$, where x was the fraction of item j assigned to i and B is the budget of i. Moreover, the portion of budget spent by i is at most $(B - bx)$. This implies $value(j) \geq bx$. The claim follows by noting for all $b \leq B$ and all x,[‡]

$$bx \geq \frac{3}{4} \cdot \frac{4bx - B}{3x}$$

In Step 3, in fact the LP drop equals the value obtained - both the LP drop and the value obtained is the sum of bids on items in $\Gamma(i)$ or B_i, whichever is less.

Coming to step 4, $Q_k = \Gamma(i) \setminus j$ be the set of goods assigned to the tight, non-lying leaf agent i. Let b and b' denote the bids of i on j before and after the step: b_{ij} and b'_{ij}. Let x be x^*_{ij}. Note that $x^*_{il} \leq 1$ for all $l \in Q_k$. Also, x^* restricted to the remaining goods still is a feasible solution in the modified instance $\mathcal{P}_{k+1}$. Since the bid on item j changes from b to b', the drop in the optimum is at most

$$LP(\mathcal{P}_k) - LP(\mathcal{P}_{k+1}) \leq (\sum_{l \in Q_k} b_{il}) + (bx - b'x)$$

Note that $value(Q_k) = \sum_{l \in Q_k} b_{il} \geq B - bx$ by tightness of i. We now show $(bx - b'x) \leq \frac{1}{3} \cdot value(Q_k)$ which would prove the lemma.

If $b' = 0$, this means $4bx \leq B$. Thus, $value(Q_k) \geq B - bx \geq 3bx$. Otherwise, we have

$$(bx - b'x) = bx - \frac{4bx - B}{3x} \cdot x = \frac{B - bx}{3} \leq value(Q_k)/3$$

implying the claim, as before. □

Remark: The algorithm above can be extended to the case of β-MBA as well. The only difference is in the modification of bids and budgets in Step 4. The modification is $b'_{ij} = \frac{4b_{ij}x^*_{ij} - \beta \cdot B_i}{(4-\beta)x^*_{ij}}$. It is not hard to modify Theorem 2.3 to show this gives a $1 - \beta/4$-factor algorithm for β-MBA.

3 Primal-dual algorithm for MBA

In this section we give a faster primal-dual algorithm for MBA although we lose a bit on the factor. The main theorem of this section is the following:

Theorem 3.1 *For any $\epsilon > 0$, there exists an algorithm which runs in $\tilde{O}(nm/\epsilon)$ time and gives a $\frac{3}{4} \cdot (1 - \epsilon)$-factor approximation algorithm for MBA.*

Let us start by taking the dual of the LP relaxation LP(1).

$$\begin{aligned} DUAL := \min\{ & \sum_{i \in A} B_i \alpha_i + \sum_{j \in Q} p_j && (2) \\ & \forall i \in A, j \in Q; \quad p_j \geq b_{ij}(1 - \alpha_i); \\ & \forall i \in A, j \in Q; \quad p_j, \alpha_i \geq 0 \} \end{aligned}$$

[‡] $4bx^2 - 4bx + B = b(2x-1)^2 + (B - b) \geq 0$

We make the following interpretation of the dual variables: Every agent retains α_i of his budget, and all his bids are modified to $b_{ij}(1-\alpha_i)$. The price p_j of a good is the highest modified bid on it. The dual program finds retention factors to minimize the sum of budgets retained and prices of items. We start with a few definitions.

Definition 2 *Let $\Gamma : A \to 2^Q$ be an allocation of items to agents and let the set $\Gamma(i)$ be called the items* owned *by i. Let $S_i := \sum_{j \in \Gamma(i)} b_{ij}$ denote the total bids of i on items in $\Gamma(i)$. Note that the revenue generated by Γ from agent i is $\min(S_i, B_i)$. Given α_i's, the* prices generated by Γ *is defined as follows: $p_j = b_{ij}(1 - \alpha_i)$, where j is owned by i. Call an item* wrongly allocated *if $p_j < b_{lj}(1 - \alpha_l)$ for some agent l, call it* rightly allocated *otherwise. An allocation Γ is called* valid *(w.r.t α_i's) if all items are rightly allocated, that is, according to the interpretation of the dual given above, all items go to agents with the highest modified bid ($b_{ij}(1 - \alpha_i)$) on it. Note that if Γ is valid, (p_j,α_i)'s form a valid dual solution. Given an $\epsilon > 0$, Γ is* ϵ-valid *if $p_j/(1 - \epsilon)$ satisfies the dual feasibility constraints with the α_i's.*

Observe that given α_i's; and given an allocation Γ and thus the prices p_j generated by it, the objective of the dual program can be treated agent-by-agent as follows

$$DUAL = \sum_i Dual(i), \quad \text{where} \qquad (3)$$

$$Dual(i) = B_i\alpha_i + \sum_{j \in \Gamma(i)} p_j = B_i\alpha_i + S_i(1 - \alpha_i)$$

Now we are ready to describe the main idea of the primal-dual schema. The algorithm starts with all α_i's set to 0 and an allocation valid w.r.t to these. We will "pay-off" this dual by the value obtained from the allocation agent-by-agent. That is, we want to pay-off $Dual(i)$ with $\min(B_i, S_i)$ for all agents i. Call an agent *paid for* if $\min(B_i, S_i) \geq \frac{3}{4} Dual(i)$. We will be done if we find α_i's and an allocation valid w.r.t these such that all agents are paid for.

Let us look at when an agent is paid for. From the definition of $Dual(i)$, an easy calculation shows that an agent is paid for iff $S_i \in [L(\alpha_i), U(\alpha_i)] \cdot B_i$, where $L(\alpha) = \frac{3\alpha}{1+3\alpha}$ and $U(\alpha) = \frac{4-3\alpha}{3-3\alpha}$. Note that S_i depends on Γ which was chosen to be valid w.r.t. α_i's. Moreover, observe that increasing α_i can only lead to the decrease of S_i and vice-versa. This suggests the following next step: for agents i which are unpaid for, if $S_i > U(\alpha_i)B_i$, increase α_i and if $S_i < L(\alpha_i)B_i$, decrease α_i and modify Γ to be the valid allocation w.r.t the α_i's.

However, it is hard to analyze the termination of an algorithm which both increases and decreases α_i's. This is where we use the following observation about the function $L()$ and $U()$. (In fact $3/4$ is the largest factor for which the corresponding $L()$ and $U()$ have the following property; see Remark 3.4)

Property 3.2 *For all α, $U(\alpha) \geq L(\alpha) + 1$.* [§]

The above property shows that an agent with $S_i > U(\alpha_i)B_i$ on losing a *single item* j will still have $S_i > U(\alpha_i)B_i - b_{ij} \geq (U(\alpha_i) - 1)B_i \geq L(\alpha_i)B_i$, for any $\alpha_i \in [0, 1]$. Also observe that in the beginning when α_i's are 0, $S_i \geq L(\alpha_i)B_i$. Thus if we can make sure that the size of $\Gamma(i)$ decreases by at most one, when the α_i's of an unpaid agent i is increased, then the case $S_i < L(\alpha_i)B_i$ never occurs and therefore we will never have to decrease α's and termination will be guaranteed.

However, an increase in α_i can lead to movement of more than one item from the current allocation of agent i to the new valid allocation. Thus to ensure *steady* progress is made throughout, we move to ϵ-valid allocations and get a $\frac{3}{4} \cdot (1-\epsilon)$ algorithm. We now give details of the Algorithm 2.

Algorithm 2 MBA-PD: Primal Dual Algorithm for MBA

Define $\epsilon_i := \epsilon \cdot \frac{1-\alpha_i}{\alpha_i}$. Throughout, p_j will be the price generated by Γ and current α_i's.

```
1. Initialize $\alpha_i = 0$ for all agents. Let $\Gamma$ be the allocation
   assigning item $j$ to agent $i$ which maximizes $b_{ij}$.

2. Repeat the following till all agents are paid for:

       Pick an agent $i$ who is not paid for (that is $S_i > U(\alpha_i)B_i$), arbitrarily. Repeat till $i$ becomes paid for:

           If $i$ has no wrongly allocated items in $\Gamma(i)$, then increase $\alpha_i \rightarrow \alpha_i(1+\epsilon_i)$. (Note that when $\alpha_i = 0$, $\epsilon_i$ is undefined. In that case, modify $\alpha_i = \epsilon$ from 0.)

           Else pick any one wrongly allocated item $j$ of agent $i$, and modify $\Gamma$ by allocating $j$ to the agent $l$ who maximizes $b_{lj}(1-\alpha_l)$. (Note that this makes $j$ rightly allocated but can potentially make agent $l$ not paid for).
```

Claim 3.3 *Throughout the algorithm, $S_i \geq L(\alpha_i)B_i$.*

Proof: The claim is true to start with ($L(0) = 0$). Moreover, S_i of an agent i decreases *only if* i is not paid for, that is, $S_i > U(\alpha_i)B_i$. Now, since items are transferred one at a time and each item can contribute at most B_i to S_i, the fact $U(\alpha) \geq 1 + L(\alpha)$ for all α proves the claim. □

[§] $U(\alpha) - 1 = \frac{1}{3-3\alpha} \geq \frac{3\alpha}{1+3\alpha} =: L(\alpha) \Leftarrow 1 + 3\alpha \geq 9\alpha(1-\alpha) \Leftarrow 9\alpha^2 - 6\alpha + 1 \geq 0$

Remark 3.4 *In general, one can compare $Dual(i)$ and $\min(S_i, B_i)$ to figure out what L, U should be to get a ρ-approximation. As it turns out, the largest ρ for which U, L satisfies property 3.2 is $3/4$ (and it cannot be any larger due to the integrality gap example). However, the bottleneck above is the fact that each item can contribute at most B_i to S_i. Note that in the case of β-MBA this is $\beta \cdot B_i$ and indeed this is what gives a better factor algorithm. See Theorem 3.8.*

Theorem 3.5 *For any $\epsilon > 0$, given α_i's, an allocation Γ ϵ-valid w.r.t it and p_j, the prices generated by Γ; if all agents are paid for then Γ is a $3/4(1-\epsilon)$-factor approximation for MBA.*

Proof: Consider the dual solution (p_j, α_i). Since all agents are paid for, $\min(B_i, S_i) \geq 3/4 \cdot Dual(i)$. Thus the total value obtained from Γ is at least $3/4 \sum_{i \in A} Dual(i)$. Moreover, since Γ is ϵ-valid, $(p_j/(1-\epsilon), \alpha_i)$ forms a valid dual of cost $\frac{1}{1-\epsilon} \sum_{i \in A} Dual(i)$ which is an upper bound on the optimum of the LP and thus the proof follows. □

Along with Theorem 3.5, the following theorem about the running time proves Theorem 3.1.

Theorem 3.6 *Algorithm* MBA-PD *terminates in $(nm \cdot \ln(3m)/\epsilon)$ iterations with an allocation Γ with all agents paid for. Moreover, the allocation is ϵ-valid w.r.t the final α_i's.*

Proof: Let us first show the allocation throughout remains ϵ-valid w.r.t. the α_i's. Note that initially the allocation is valid. Subsequently, the price of an item j generated by Γ decreases only when the α_i of an agent i owning j increases. This happens only in Step 2, and moreover j must be rightly allocated before the increase. Now the following calculation shows that after the increase of α_i, p_j decreases by a factor of $(1-\epsilon)$. Thus, $(p_j/(1-\epsilon), \alpha_i)$'s form a valid dual solution implying Γ is ϵ-valid.

$$\begin{aligned} p_j^{(new)} &= b_{ij}(1 - \alpha_i^{(new)}) = b_{ij}(1 - \alpha_i(1+\epsilon_i)) \\ &= b_{ij}(1-\alpha_i)(1 - \epsilon_i\alpha_i/(1-\alpha_i)) = p_j^{(old)}(1-\epsilon) \end{aligned}$$

Now in Step 2, note that until there are agents not paid for, either we decrease the number of wrongly allocated items or we increase the α_i for some agent i. That is, in at most m iterations of Step 2, α_i of some agent becomes $\alpha_i(1+\epsilon_i)$. Now, note that if $\alpha_i > 1 - 1/3m$ for some agent, he is paid for. This follows simply by noting that $S_i \leq mB_i = U(1 - 1/3m) \cdot B_i$ and the fact that $S_i \geq L(\alpha_i)B_i$, for all α_i.

Claim 3.7 *If α_i is increased $t > 0$ times, then it becomes $1 - (1-\epsilon)^t$.*

Proof: At $t = 1$, the claim is true as α_i becomes ϵ. Suppose the claim is true for some $t \geq 1$. On the $t + 1$th increase, α_i goes to

$$\begin{aligned}\alpha_i(1+\epsilon_i) &= \alpha_i + \epsilon(1-\alpha_i) = \alpha_i(1-\epsilon)+\epsilon \\ &= (1-(1-\epsilon)^t)(1-\epsilon)+\epsilon \\ &= 1-(1-\epsilon)^{t+1}\end{aligned}$$

□

Thus if α_i is increased $\ln(3m)/\epsilon$ times, i becomes paid for throughout the remainder of the algorithm. Since there are n agents, and in each m-steps some agent's α_i increases, in $(nm \cdot \ln(3m)/\epsilon)$ iterations all agents are paid for and the algorithm terminates. □

The algorithm can be generalized for β-MBA by changing the definition of *paid for* and $L(), U()$. Call an agent paid for if $\min(B_i, S_i) \geq \frac{4-\beta}{4} Dual(i)$. Define the function $L(\alpha) := \frac{\alpha(4-\beta)}{\alpha(4-\beta)+\beta}$ and $U(\alpha) := \frac{(1-\alpha)(4-\beta)+\beta}{(1-\alpha)(4-\beta)}$ Note that when $\beta = 1$, the definitions coincide with the definitions in the previous section. The following theorem generalizes Theorem 3.1.

Theorem 3.8 *The algorithm 2 with the above definitions gives a $(1-\beta/4)(1-\epsilon)$-factor approximation for β-MBA in $\tilde{O}(nm/\epsilon)$ time.*

4 Inapproximability of MBA and related problems

In this section we study the inapproximability of MBA and the related problems as stated in the introduction. The main theorem of this section is the following 15/16 hardness of approximation factor for MBA.

Theorem 4.1 *For any $\epsilon > 0$, it is NP-hard to approximate* MBA *to a factor $15/16 + \epsilon$. This holds even for* uniform *instances.*

We give a reduction from MAX-3-LIN(2) to MBA to prove the above theorem. The MAX-3-LIN(2) problem is as follows: Given a set of m equations in n variables over $GF(2)$, where each equation contains exactly 3 variables, find an assignment to the variables to maximize the number of satisfied equations. Håstad, in his seminal work [13], gave the following theorem.

Theorem 4.2 *[13] Given an instance I of* MAX-3-LIN(2)*, for any $\delta, \eta > 0$, its NP hard to distinguish between the two cases:* YES*: There is an assignment satisfying $(1-\delta)$-fraction of equations, and* NO*: No assignment satisfies more than $(1/2+\eta)$-fraction of equations.*

Let I be an instance of MAX-3-LIN(2). Denote the variables as $x_1, \cdots, x_n$. Also let $deg(x_i)$ be the *degree* of variable x_i i.e. the number of equations in which variable x_i occurs. Note that $\sum_i deg(x_i) = 3m$. We construct an instance $R(I)$ of MBA as follows:

- For every variable x_i, we have two agents which we label as $\langle x_i : 0\rangle$ and $\langle x_i : 1\rangle$, corresponding to the two assignments. The budget of both these agents is $4deg(x_i)$ (4 per equation).

- There are two kinds of items. For every variable x_i, we have a *switch item* s_i. Both agents, $\langle x_i : 0\rangle$ and $\langle x_i : 1\rangle$, bid their budget $4deg(x_i)$ on s_i. No one else bids on s_i.

- For every equation $e : x_i + x_j + x_k = \alpha$ ($\alpha \in \{0, 1\}$), we have 4 kinds of items corresponding to the four assignments to x_i, x_j, x_k which satisfy the equation: $\langle x_i : \alpha, x_j : \alpha, x_k : \alpha\rangle$, $\langle x_i : \alpha, x_j : \bar{\alpha}, x_k : \bar{\alpha}\rangle$, $\langle x_i : \bar{\alpha}, x_j : \bar{\alpha}, x_k : \alpha\rangle$ and $\langle x_i : \bar{\alpha}, x_j : \alpha, x_k : \bar{\alpha}\rangle$. For each equation, we have 3 copies of each of the four items. The set of all 12 items are called *equation items*, and denoted by S_e. Thus we have $12m$ equation items, in all.

  For every equation item of the form $\langle x_i : \alpha_i, x_j : \alpha_j, x_k : \alpha_k\rangle$, only three agents bid on it: the agents $\langle x_i : \alpha_i\rangle$, $\langle x_j : \alpha_j\rangle$ and $\langle x_k : \alpha_k\rangle$. The bids are of value 1 each.

The following lemma is not hard to see.

Lemma 4.3 *There always exists an optimal solution to $R(I)$ in which every switch item is allocated.*

Since agents who get switch items exhaust their budget, any more equation items given to them generate no extra revenue. We say that an equation item can be allocated in $R(I)$ only if it generates revenue, that is, it is not allocated to an agent who has spent all his budget.

Lemma 4.4 *Given an assignment of variables by $R(I)$, if an equation e is satisfied then all the 12 items of S_e can be allocated in $R(I)$. Otherwise, at most 9 items of S_e can be allocated in $R(I)$.*

Proof: If an equation e is satisfied, then there must be one equation item $\langle x_i : \alpha_i, x_j : \alpha_j, x_k : \alpha_k\rangle$ such that x_r is assigned α_r ($r = i, j, k$) in the assignment by $R(I)$ (that is the switch item s_r is given to $\langle x_r : \bar{\alpha_r}\rangle$). Assign the 12 items of S_e as follows: give one the three copies of $\langle x_i : \alpha_i, x_j : \alpha_j, x_k : \alpha_k\rangle$ to agents $\langle x_i : \alpha_i\rangle$, $\langle x_j : \alpha_j\rangle$ and $\langle x_k : \alpha_k\rangle$. Note that none of them have got the switch item. Moreover, for the other items in S_e, give all 3 copies of $\langle x_i : \alpha_i, x_j : \bar{\alpha_j}, x_k : \bar{\alpha_k}\rangle$ to agent $\langle x_i : \alpha_i\rangle$, and similarly for the three copies of $\langle x_i : \bar{\alpha_i}, x_j : \alpha_j, x_k : \bar{\alpha_k}\rangle$ and $\langle x_i : \bar{\alpha_i}, x_j : \bar{\alpha_j}, x_k : \alpha_k\rangle$. Since each agent gets 4 items, he does not exhaust his budget.

If an equation e is *not* satisfied, then observe that there must be an equation item $\langle x_i : \alpha_i, x_j : \alpha_j, x_k : \alpha_k\rangle$ such that x_r is assigned $\bar{\alpha_r}$ $(r = i, j, k)$ in the assignment. That is, all the three agents bidding on this item have their budgets filled up via switch items. Thus none of the copies of this equation item can be allocated, implying at most 9 items can be allocated. □

The following two lemma along with Håstad's theorem prove the hardness for maximum budgeted allocation given in Theorem 4.1.

Lemma 4.5 *If $OPT(I) \geq m(1 - \epsilon)$, then the maximum budgeted allocation revenue of $R(I)$ is at least $24m - 12m\epsilon$.*

Proof: Allocate the switch elements in $R(I)$ so that the $assignment$ of variables by $R(I)$ is same as the $assignment$ of I. That is, if x_i is assigned 1 in the solution to I, allocate s_i to $\langle x_i : 0\rangle$, and vice versa if x_i is assigned 0. For every equation which is satisfied, allocate the 12 equation items as described in Lemma(4.4). Since each agent gets at most 4 items per equation, it gets at most $4deg(x_i)$ revenue which is under his budget. Thus the total budgeted allocation gives revenue: gain from switch items + gain from equation items $= \sum_i 4deg(x_i) + 12m(1 - \epsilon)$ = $24m - 12m\epsilon$. □

Lemma 4.6 *If $OPT(I) \leq m(1/2 + \eta)$, then the maximum budgeted allocation revenue of $R(I)$ is at most $22.5m + 3m\eta$*

Proof: Suppose not. i.e . the maximum revenue of $R(I)$ is strictly greater than $22.5m + 3m\eta$. Since the switch items can attain at most $12m$ revenue, $10.5m + 3m\eta$ must have been obtained from equation items. We claim that there must be strictly more than $m(1/2 + \eta)$ equations so that at least 10 out of their 12 equation items are allocated. Otherwise the revenue generated will be at most $12m(1/2 + \eta) + 9m(1/2 - \eta) = 10.5m + 3m\eta$ The contradiction follows from Lemma(4.4). □

As noted in Section 1.2, MBA is a special case of SMW. Thus the hardness of approximation in Theorem 4.1 would imply a hardness of approximation for SMW with the demand oracle, if the demand oracle could be simulated in poly-time in the hard instances of MBA. Lemma 4.8 below shows that this indeed is the case which gives the following theorem.

Theorem 4.7 *For any $\epsilon > 0$, it is NP-hard to approximate submodular welfare with demand queries to a factor $15/16 + \epsilon$.*

Lemma 4.8 *Given any instance I of* MAX-3-LIN(2)*, in the corresponding instance $R(I)$ as defined in Section 4 the demand oracle can be simulated in polynomial time.*

Proof: We need to show that for any agent i and given prices $p_1, p_2 \cdots$ to the various items, one can find a subset of items S which maximizes $(\min(B_i, \sum_{j\in S} b_{ij}) - \sum_{j\in S} p_j)$. Call such a bundle the optimal bundle. Observe that in the instance $R(I)$, the bid of an agent i is 1 on an equation item and B_i on the switch item. Therefore, the optimal bundle S either consists of just the switch item or consists of B_i equation items. The best equation items are obviously those of the smallest price and thus can be found easily (in particular in polynomial time). □

The following hardness of approximation results for β-MBA, GAP and weighted MSSF are obtained using similar reductions and are deferred to in the full version of the paper.

Theorem 4.9 *For any $\epsilon > 0$, it is NP-hard to approximate β-MBA to a factor $1 - \beta/16 + \epsilon$.*

Theorem 4.10 *For any $\epsilon > 0$, it is NP-hard to approximate GAP to a factor $10/11 + \epsilon$.*

Theorem 4.11 *For any $\epsilon > 0$, it is NP-hard to approximate edge-weighted MSSF and node-weighted to a factor $10/11 + \epsilon$ and $13/14 + \epsilon$ respectively.*

5 Discussion

In this paper we studied the maximum budgeted allocation problem and showed that the true approximability lies between $3/4$ and $15/16$. Our algorithms were based on a natural LP relaxation of the problem and we, in some sense, got the best out if it: the integrality gap of the LP is $3/4$. An approach to get better approximation algorithms might be to look at stronger LP relaxations of the problem. One such relaxation is the *configurational LP relaxation*. Configurational LPs have been used for other allocation problems; in fact the best known approximation algorithm of Feige and Vondrák [10] for SMW and GAP proceeds by rounding the solution of the LP. In the full version of the paper, we demonstrate an example which shows that the integrality gap of this LP is at most $5/6$. We leave the pinning down of the integrality gap as an open question and we believe that the resolution might require some newer techniques in addition to the ones developed in this paper.

Another direction of research is improving the hardness of approximation factor. We define a natural extention of SAT below which seems to be at the core of MBA and other allocation problems like GAP and SWM.

Definition 3 BUDGETED-3SAT(b)*: Given a formula* $\phi = C_1 \wedge C_2 \cdots \wedge C_m$*, where each clause* C_i *is a disjunction of three literals, let the degree,* $deg(l)$*, of a literal* l *be the number of clauses the literal appears in. The* BUDGETED-3SAT(b) *asks for an assignment of variables and a maximum sized pairing* $\{l_i, C_i\}$ *for every clause* C_i *and* $l_i \in C_i$ *such that the assignment of the literal satisfies the disjunction* C_i*, and every literal* l *appears in at most* $b \cdot deg(l)$ *pairings.*

Clearly, MAX-3-SAT is just the BUDGETED-3SAT(1) problem. Our hardness result essentially shows that BUDGETED-3SAT(b) (for any budget b) can be reduced to MBA, and a hardness of $\rho(b)$ for BUDGETED-3SAT(b) implies a hardness of $\frac{2\rho(b)+3b}{2+3b}$. Implicit in our reduction is the fact that BUDGETED-3SAT(2/3) is 7/8-hard which gives us a $15/16$ hardness for MBA.

For general values of b however the approximibility BUDGETED-3SAT(b) is open. In fact, for general b, it is not even clear if BUDGETED-3SAT(b) is easier or harder than MAX-3-SAT. Obtaining hardness results for BUDGETED-3SAT(b) seems to be an interesting approach towards the inapproximibility of allocation problems.

Acknowledgements: We would like to thank Anna Karlin, Aranyak Mehta and Vijay Vazirani for helpful suggestions.

References

[1] N. Andelman. *Online and strategic aspects of network resource management algorithms*. PhD thesis, Tel Aviv University, Tel Aviv, Israel, 2006.

[2] N. Andelman and Y. Mansour. Auctions with budget constraints. *Proceedings of SWAT*, 2004.

[3] Y. Azar, B. Birnbaum, A. Karlin, C. Mathieu, and C. Nguyen. Improved approximation algorithms for budgeted allocations. *Proceedings of ICALP*, pages 186–197, 2008.

[4] J.-P. Benoit and V. Krishna. Multiple object auctions with budget constrained bidders. *Review of Economic Studies*, 68:155–179, 2001.

[5] L. Blumrosen and N. Nisan. Combinatorial auctions. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 267–299. Cambridge University Press, 2007.

[6] N. Buchbinder, K. Jain, and S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. *Proceedings of ESA*, 2007.

[7] C. Chekuri and S. Khanna. A PTAS for the multiple-knapsack problem. *Proceedings of SODA*, 2000.

[8] N. Chen, R. Engelberg, C. Nguyen, P. Raghavendra, A. Rudra, and G. Singh. Improved approximation algorithms for the spanning star forest problem. *Proceedings of APPROX*, 2007.

[9] M. Chlebik and J. Chlebikova. Approximation hardness for small occurrence instances of np-hard problems. *Proceedings of CIAC*, 2003.

[10] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. *Proceedings of FOCS*, 2006.

[11] L. Fleischer, M. X. Goemans, V. Mirrokini, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems,. *Proceedings of SODA*, 2006.

[12] R. Garg, V. Kumar, and V. Pandit. Approximation algorithms for budget-constrained auctions. *Proceedings of APPROX*, 2001.

[13] J. Håstad. Some optimal inapproximability results. *JACM*, 48:798–859, 2001.

[14] K. Jain. Factor 2 approximation algorithm for the generalized steiner network problem. *Proceedings of FOCS*, 1998.

[15] S. Khot, R. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Proceedings of WINE*, 2005.

[16] L. C. Lau, S. Naor, M. Salavatipour, and M. Singh. Survivable network design with degree or order constraints. *Proceedings of STOC*, 2007.

[17] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Proceedings of EC*, 2001.

[18] J. K. Lenstra, D. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46:259–271, 1990.

[19] A. Mehta, A. Saberi, U. V. Vazirani, and V. V. Vazirani. Adwords and generalized on-line matching. *Proceedings of FOCS*, 2005.

[20] C. Nguyen, J. Shen, M. M. Hou, L. Sheng, W. Miller, and L. Zhang. Approximating the spanning star forest problem and its applications to genomic sequence alignment. *Proceedings of SODA*, 2007.

[21] T. Sandholm and S. Suri. Side constraints and non-price attributes in markets. *Proceedings of IJCAI*, 2001.

[22] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Math. Programming*, 62:461–474, 1993.

[23] A. Srinivasan. Budgeted allocations in the full-information setting. *To appear in Proceedings of APPROX*, 2008.

[24] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. *Proceedings of STOC*, 2008.