

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 $\simeq 0.632$ [AM04, FV06]. 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 [SS06, LLN01]. 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 $275/276$ [FV06].
- 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 [CK00, CC02].

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 [BK01]) 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).

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 $\Gamma : A \rightarrow 2^Q$ is the partitioning the sets of items Q into disjoint sets $\Gamma(1), \dots, \Gamma(n)$. 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 ([GKP01, SS06, AM04, LLN01]). The first approximation algorithm for the problem was given by Garg, Kumar and Pandit[GKP01] who gave a $2/(1 + \sqrt{5}) (\simeq 0.618)$ factor approximation. Andelman and Mansour[AM04] improved the factor to $(1 - 1/e) (\simeq 0.632)$. For the special case when budgets of all bidders were equal, [AM04] improved the factor to 0.717. We refer to the thesis of Andelman[And06] for an exposition. Very recently, and independent of our work, Azar et.al. [ABK⁺08] obtained a 2/3-factor for the general MBA problem. They also considered a *uniform* version of the problem where for every item j , the bid of any agent is either b_j (independent of the agent) or 0. They gave a $1/\sqrt{2} (\simeq 0.707)$ factor for the same. All these algorithms are based on a natural LP relaxation (LP(1) in Section 1.3) which we use as well.

In the setting of sponsored search auctions, MBA, or rather β -MBA with $\beta \rightarrow 0$, has been studied mainly in an online context. Mehta et.al.[MSVV07] and later, Buchbinder et.al.[BJN07] 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.

*see for instance <http://www.google.com/adwords/audioads/> and <http://www.google.com/adwords/tvads/>

[†]PARTITION: Given n integers a_1, \dots, a_n and a target B , decide whether there is a subset of these integers adding up to exactly B

Our results: We give two approximation algorithms for MBA. The first, based on iterative LP rounding, attains a factor of $3/4$. The algorithm described in Section 2. The second algorithm, based on the primal-dual schema, is faster and attains a factor of $3/4(1 - \epsilon)$, for any $\epsilon > 0$. The running time of the algorithm is $\tilde{O}(\frac{nm}{\epsilon})^\ddagger$, and is thus almost linear for constant ϵ and dense instances. 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. [ABK⁺08] 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 $u_i : 2^Q \rightarrow \mathbb{R}_+$ 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.

Before discussing the complexity of the welfare maximization problem, one needs to be careful of how the utility functions are represented. Since it takes exponential (in the number of items) size to represent a general set-function, *oracle access* to these functions are assumed and the complexity of the welfare maximization problem depends on the strength of the oracle. The strongest such oracle that has been studied is the so-called *demand oracle*: for any agent i and prices p_1, p_2, \dots, p_m for all items in Q , returns a subset $S \subseteq Q$ which maximizes $(u_i(S) - \sum_{j \in S} p_j)$.

Welfare maximization problems have been extensively studied (see, for example, [BN07]) 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 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. Feige and Vondrák [FV06] gave an $(1 - 1/e + \rho)$ -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 $u_i(S) = \min(B_i, \sum_{j \in S} b_{ij})$ when B_i, b_{ij} 's are fixed is a submodular function. In Section 4.1, 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 as well.

Generalized Assignment Problem (GAP): GAP is a problem quite 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 $\sum_{j \in S} s_{ij} \leq C_i$. The goal is to find a revenue (profit) maximizing feasible

[‡]the \sim hides logarithmic factors

[§]We remark that SWM with a different oracle, the *value oracle* which given a set and an agent returns the utility of the agent for the set, has recently been resolved. There was a $(1 - 1/e)$ hardness given by Khot et.al.[KLMM05] and recently Vondrák[Von08] gave a matching polynomial time algorithm.

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[CK00], a $1/2$ approximation algorithm was implicit in the work of Shmoys and Tardos[ST93]. The factor was improved by Fleischer et.al.[FGMS06] to $1 - 1/e$. In the same paper [FV06] where they give the best known algorithm for SWM, Feige and Vondrák[FV06] also give a $(1 - 1/e + \rho')$ algorithm for GAP ($\rho' \leq 10^{-5}$). The best known hardness for GAP was $1 - \epsilon$, for some small ϵ which was given by Chekuri and Khanna [CK00] via a reduction from maximum 3D-matching. Improved hardness results for maximum 3D matching by Chlebik and Chlebikova[CC02], 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 in Section 4.3.

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 [NSH⁺07] 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. [CEN⁺07] 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 technique can be used to improve the hardness of node-weighted and edge-weighted MSSF to $13/14$ and $10/11$, respectively. We describe this in Section 4.4.

1.3 The LP Relaxation for MBA

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

$$\max\left\{\sum_{i \in A} \pi_i : \pi_i = \min\left(B_i, \sum_{j \in Q} b_{ij}x_{ij}\right), \forall i; \sum_{i \in A} x_{ij} \leq 1, \forall j; x_{ij} \in \{0, 1\}\right\}$$

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, $B_i \geq \sum_{j \in Q} b_{ij}x_{ij}$. This was noted by Andelman and Mansour[AM04] and a similar relaxation was used by Garg et.al. [GKP01].

$$\max\left\{\sum_{i \in A, j \in Q} b_{ij}x_{ij} : \forall i \in A, \sum_{j \in Q} b_{ij}x_{ij} \leq B_i; \forall j \in Q, \sum_{i \in A} x_{ij} \leq 1; \forall i \in A, j \in Q, x_{ij} \geq 0\right\} \tag{1}$$

We remark that the assumption $b_{ij} \leq B_i$ 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.

Moreover, the integrality gap of this LP is at most $3/4$. In the following example in Figure(1), the maximum revenue obtained by any feasible allocation is 3 while the value of the LP is 4. The example is due to [AM04] and thus our main result shows that the integrality gap is exactly $3/4$.

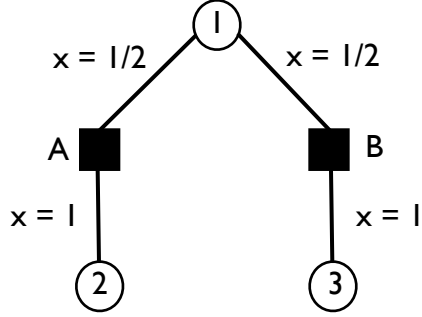


Figure 1: Agents are black squares and have budget 2. The bids of agent A and B on item 1 is 2. A bids 1 on 2 and B bids 1 on 3. Note the agent who doesn't get item 1 will spend only 1 and thus the maximum allocation is 3. The LP however gets 4 as shown by the solution x .

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 [LST90]. A similar claim can be found in the thesis of Andelman [And06].

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.*

Proof: Consider the graph $G[E^*]$. Without loss of generality assume that it is a single connected component. Otherwise we can treat every connected component as a separate instance and argue on each of them separately. Thus, $G[E^*]$ has $(n+m)$ nodes. Also since there are $(n+m)$ constraints in the LP which are not non-negativity constraints, therefore support of any extreme point solution can be of size at most $(n+m)$. This follows from simple polyhedral combinatorics: at an extreme point, the number of inequalities going tight is at least the number of variables. Since there are only $(n+m)$ constraints which are not non-negativity constraints, all but at most $(n+m)$ variables must satisfy the non-negativity constraints with equality, that is, should be 0. Thus $|E^*| \leq n+m$.

Hence there is at most one cycle in $G[E^*]$. Suppose the cycle is: $(i_1, j_1, i_2, j_2, \dots, j_k, i_1)$, where $\{i_1, \dots, i_k\}$ and $\{j_1, \dots, j_k\}$ are the subsets of agents and items respectively. Consider the feasible fractional solution obtained by decreasing x^* on (i_1, j_1) by ϵ_1 and increasing on (i_2, j_1) by ϵ_1 , decreasing on (i_2, j_1) by ϵ_2 , increasing on (i_2, j_1) by ϵ_2 , and so on. Note that if the ϵ_i 's are small enough, the item constraints are satisfied. The relation between ϵ_1 and ϵ_2 (and cascading to other ϵ_r 's) is: $\epsilon_1 b_{i_2, j_1} = \epsilon_2 b_{i_2, j_2}$, that is, the fraction of money spent by i_2 on j_1 equals the

money freed by j_2 . The exception is the last ϵ_k , which might not satisfy the condition with ϵ_1 . If $\epsilon_k b_{i_1, j_k} > \epsilon_1 b_{i_1, j_1}$, then just stop the increase on the edge (i_1, j_k) to the point where there is equality. If $\epsilon_k b_{i_1, j_k} < \epsilon_1 b_{i_1, j_1}$, then start the whole procedure by increasing x^* on the edge (i_1, j_1) instead of decreasing and so on. In one of the two cases, we will get a feasible solution of equal value and the ϵ_i 's can be so scaled so as to reduce x^* on one edge to 0. In other words, the cycle is broken without decreasing the LP value.

Thus, $G[E^*]$ is a tree. Moreover, since $(n + m - 1)$ edges are positive, there must be $(n + m - 1)$ equalities among the budget and the item constraints. Thus at most one budget constraint can be violated which implies at most one agent can *non tight*. Now since the bids are less than the budget, therefore if an agent is a leaf of the tree $G[E^*]$ then he must be *non-tight*. Hence at most one agent can be a leaf of the tree $G[E^*]$. \square

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 at most 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 (Figure 2 below) 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

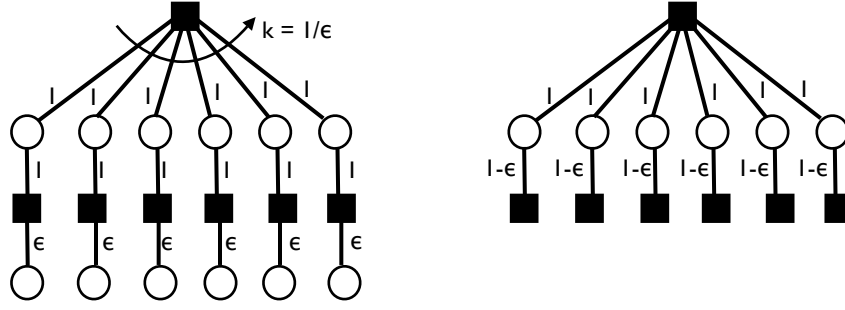


Figure 2: For some $\epsilon > 0$ let $k = 1/\epsilon$. In the instance, there are $k + 1$ agents with budgets 1 denoted by black squares and $2k$ items with bids as shown on the figure in the left. The LP value of this is $k + 1$: the ϵ edges have $x = 1$, the 1 edges forming a matching have $x = 1 - \epsilon$ and the rest have $x = \epsilon$. After the first iteration, the leaf items are assigned and the value obtained is $k\epsilon = 1$. The budgets of the k agents at the bottom reduce to $1 - \epsilon$ and so do their modified bids, as shown on the figure in the right. The LP solution for this instance is $k - 1 + \epsilon$ ($k - 1$ items going to bottom $k - 1$ agents and the remaining item to the top guy). The LP drop is $2 - \epsilon$ and thus is twice the value obtained as $\epsilon \rightarrow 0$.

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).

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 more than 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. \square

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

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

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

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.

$\sum_{j \in Q} \text{value}(j) = \sum_k (\sum_{j \in Q_k} \text{value}(j))$. 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. \square

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

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 $\text{value}(j) \geq bx$. 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

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

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

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

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. \square

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).

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

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 $\Gamma : A \rightarrow 2^Q$ 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

$$DUAL = \sum_i Dual(i), \quad \text{where } Dual(i) = B_i \alpha_i + \sum_{j \in \Gamma(i)} p_j = B_i \alpha_i + S_i(1 - \alpha_i) \quad (3)$$

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 below).

Property 3.2 For all α , $U(\alpha) \geq L(\alpha) + 1$. \parallel

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 Γ 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 $\alpha_i = 0$, ϵ_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 Γ 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. \square

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

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. Details in Section 3.1.

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. \square

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

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

\square

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. \square

3.1 Extension to β -MBA

The algorithm for β -MBA is exactly the same as Algorithm 2. The only difference is the definition of *paid for* and $L(), U()$. Call an agent paid for if $\min(B_i, S_i) \geq \frac{4-\beta}{4} \text{Dual}(i)$. 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.

Claim 3.8 *Given α_i 's, agent is paid for if $S_i \in [L(\alpha_i), U(\alpha_i)] \cdot B_i$*

Proof: Agent i is paid for if both $B_i \geq \frac{(4-\beta)}{4}(B_i\alpha_i + S_i(1-\alpha_i))$ and $S_i \geq \frac{(4-\beta)}{4}(B_i\alpha_i + S_i(1-\alpha_i))$. Let us lose the subscript for the remainder of the proof.

The first implies

$$S(1-\alpha) \leq B\left(\frac{4}{(4-\beta)} - \alpha\right) \Rightarrow S(1-\alpha) \leq B\frac{(4-\beta)(1-\alpha) + \beta}{(4-\beta)} \Rightarrow S \leq U(\alpha)B$$

The second implies

$$S\left(\frac{4}{(4-\beta)} - (1-\alpha)\right) \geq B\alpha \Rightarrow S\frac{\alpha(4-\beta) + \beta}{(4-\beta)} \geq B\alpha \Rightarrow S \geq L(\alpha)B$$

□

Property 3.9 *For all α , $U(\alpha) \geq L(\alpha) + \beta$*

Proof: Note that $U(\alpha) = 1 + \frac{\beta}{(1-\alpha)(4-\beta)}$ and $L(\alpha) = 1 - \frac{\beta}{\alpha(4-\beta)+\beta}$. Now,

$$\begin{aligned} U(\alpha) - \beta &\geq L(\alpha) \Leftrightarrow \frac{\beta}{(1-\alpha)(4-\beta)} - \beta \geq \frac{\beta}{\alpha(4-\beta) + \beta} \\ &\Leftrightarrow \frac{1}{(1-\alpha)(4-\beta)} \geq \frac{\alpha(4-\beta) - (1-\beta)}{\alpha(4-\beta) + \beta} \\ &\Leftrightarrow \alpha(4-\beta) + \beta \geq (4-\beta)^2\alpha(1-\alpha) - (1-\alpha)(1-\beta)(4-\beta) \\ \Leftrightarrow \alpha^2(4-\beta)^2 - \alpha(4-\beta)((1-\beta) + (4-\beta) - 1) + \beta + (1-\beta)(4-\beta) &\geq 0 \\ &\Leftrightarrow (\alpha(4-\beta))^2 - 2\alpha(4-\beta)(2-\beta) + (2-\beta)^2 \geq 0 \\ &\Leftrightarrow (\alpha(4-\beta) - (2-\beta))^2 \geq 0 \end{aligned}$$

which is true for any α . □

Theorem 3.10 *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.*

Proof: Armed with the Property 3.9 which implies $S_i \geq L(\alpha)B_i$ for all i , the proof of Theorem 3.6 can be modified (the only difference is we need to run till $\alpha_i > 1 - 1/(4-\beta)m$ instead of $(1 - 1/3m)$), to show that the algorithm terminates with an ϵ -valid allocation with all agents paid for. The proof of the factor follows from the proof of Theorem 3.5 and Claim 3.8. □

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 [Hås01], gave the following theorem.

Theorem 4.2 [Hås01] *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.*

We now describe the main idea of the hardness reduction, the same idea will also be used in the reduction for other problems. For every variable x in an MAX-3-LIN(2) instance, we will have two agents corresponding to the variable being 0 or 1. For each such pair of agents we have a *switch item*, an item bid on only by this pair of agents, and the allocation of the item will coincide with the assignment of the variable. For every equation e in the MAX-3-LIN(2) instance, we will have items coinciding with the satisfying assignments of the equation. For instance if the equation $e : x + y + z = 0$, we will have items corresponding to $\langle x : 0, y : 0, z : 0 \rangle$, $\langle x : 0, y : 1, z : 1 \rangle$ and so on. Each such item will be desired by the three corresponding agents: for example $\langle x : 0, y : 0, z : 0 \rangle$ will be wanted by the 0 agent corresponding to x, y and z . The bids and budgets are so set so that the switch items are always allocated and thus each allocation corresponds to an assignment. In this way, an allocation instance encodes an assignment instance. The hardness of MBA and other allocation problems follows from the hardness of MAX-3-LIN(2). We give the details now.

Let I be an instance of MAX-3-LIN(2). Denote the variables as x_1, \dots, 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.

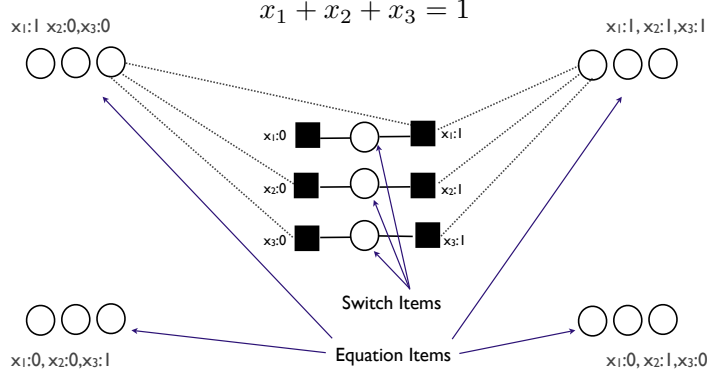


Figure 3: The hardness gadget for reduction of MBA to MAX-3-LIN(2). Dotted lines are a bid of 1 and the solid lines are a bid equaling the budget, $4deg(x_i)$.

Figure 3 illustrates the reduction above locally on three variables x_1, x_2, x_3 for the equation $x_1 + x_2 + x_3 = 1$.

We call a solution to $R(I)$ a *valid* assignment if it allocates all the switch items. 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, that is the solution is valid.*

Proof: Suppose there is a solution which is not valid. Thus there is a switch item s_i which is not allocated. Allocating s_i to either $\langle x_i : 0 \rangle$ or $\langle x_i : 1 \rangle$ and de-allocating the items allocated to the agent can only increase the value of the allocation. \square

Suppose $R(I)$ allocates switch item s_i to agent $\langle x_i : 0 \rangle$, then we say that $R(I)$ assigns x_i to 1, and similarly if s_i is allocated to $\langle x_i : 1 \rangle$ then we say x_i is assigned to 0. Thus by lemma 4.3, every optimal solution of $R(I)$ also gives an assignment of variables for I , and we call this the assignment by $R(I)$. Now observe the following property which is used to prove a crucial lemma 4.5:

Property 4.4 *If $(x_i = \alpha_i, x_j = \alpha_j, x_k = \alpha_k)$ is a satisfying assignment for the equation $x_i + x_j + x_k = \alpha$, then the other three satisfying assignments are $(x_i = \bar{\alpha}_i, x_j = \bar{\alpha}_j, x_k = \alpha_k)$, $(x_i = \bar{\alpha}_i, x_j = \alpha_j, x_k = \bar{\alpha}_k)$, and $(x_i = \alpha_i, x_j = \bar{\alpha}_j, x_k = \bar{\alpha}_k)$.*

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 4.5 *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. \square

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

Lemma 4.6 *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(4.5). 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 = $\sum_i 4deg(x_i) + 12m(1 - \epsilon) = 24m - 12m\epsilon$. \square

Lemma 4.7 *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(4.5). \square

4.1 Hardness of SMW with demand oracle

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 4.9 below shows that this indeed is the case which gives the following theorem.

Theorem 4.8 *For any $\epsilon > 0$, it is NP-hard to approximate submodular welfare with demand queries to a factor $15/16 + \epsilon$.*

Lemma 4.9 *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 $p_1, p_2 \dots$ 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). \square

4.2 Hardness of β -MBA

The hardness reduction given above can be easily modified to give a hardness result for β -MBA, for any constant $1 \geq \beta > 0$. Note that the budget of an agent is four times the degree of the analogous variable. We increase the budget of agents $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$ to $\frac{1}{\beta}4deg(x_i)$. For each agent, introduce dummy items so that the total bid of an agent on these dummy items is $(\frac{1}{\beta} - 1)$ times its original budget. The rest of the reduction remains the same. Call this new instance β - $R(I)$.

Claim 4.10 *We can assume that in any optimal allocation, all the dummy items are assigned*

Proof: If a dummy item is not assigned and assigning it exceeds the budget of the agent implies the agent must be allocated an equation item. De-allocating the equation item and allocating the dummy item gives an allocation of at least the original cost. \square

Once the dummy items are assigned, the instance reduces to the original instance. We have the following analogous lemmas of Lemma4.6 and Lemma4.7.

Lemma 4.11 *If $OPT(I) \geq m(1 - \epsilon)$, then the maximum budgeted allocation revenue of β - $R(I)$ is at least $24m - 12m\epsilon + 24m(\frac{1}{\beta} - 1)$. If $OPT(I) \leq m(1/2 + \eta)$, then the maximum budgeted allocation revenue of $R(I)$ is at most $22.5m + 3m\eta + 24m(\frac{1}{\beta} - 1)$*

Proof: The extra $24m(\frac{1}{\beta} - 1)$ is just the total value of the dummy items which is obtained in both cases. \square

The above theorem with Håstad's theorem gives the following hardness result for β -MBA.

Theorem 4.12 *For any $\epsilon > 0$, it is NP-hard to approximate β -MBA to a factor $1 - \beta/16 + \epsilon$.*

4.3 Hardness of GAP

To remind, in the generalized assignment problem (GAP) we have n bins each with a capacity B_i . There are a set of items with item j having a profit p_{ij} and size s_{ij} corresponding to bin i . The objective is to find an allocation of items to bins so that no capacities are violated and the total profit obtained is maximized.

One of the bottlenecks for getting a better lower bound for MBA is the extra contribution of switch items which are always allocated irrespective of I . A way of decreasing the effect of these switch items is to decrease their value. In the case of MBA this implies reducing the bids of agents on switch items. Note that this might lead to an agent having a switch item and an equation item as he has budget remaining, and thus the allocation does not correspond to an assignment for the variables. This is where the generality of GAP helps us: the switch item will have a reduced profit but the size will still be the capacity of the agent (bin). However, since we would want switch items to be *always* allocated, we cannot reduce their profits by too much. We use this idea to get the following theorem.

Theorem 4.13 *For any $\epsilon > 0$, it is NP-hard to approximate GAP to a factor $10/11 + \epsilon$.*

We now describe our gadget more in detail. The gadget is very much like the one used for MBA.

- For every variable x_i , we have two bins $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$, corresponding to the two assignments. The capacity of both these bins is $2deg(x_i)$ (2 per equation).

- There are two kinds of items. For every variable x_i , we have a *switch item* s_i . s_i can go to only one of the two bins, $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$. Its capacity for both bins is $2deg(x_i)$ while its profit is $deg(x_i)/2$.
- For every equation of the form $e : x_i + x_j + x_k = \alpha$ ($\alpha \in \{0, 1\}$), we have a set S_e of 4 items, called *equation 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 $\langle x_i : \bar{\alpha}, x_j : \alpha, x_k : \bar{\alpha} \rangle$. Thus we have $4m$ equation items, in all. Every equation item of the form $\langle x_i : \alpha_i, x_j : \alpha_j, x_k : \alpha_k \rangle$, can go to any one of the three bins $\langle x_i : \alpha_i \rangle$, $\langle x_j : \alpha_j \rangle$ and $\langle x_k : \alpha_k \rangle$. The profit and size for each of this bins is 1.

We will use $R(I)$ to refer to the instance of GAP obtained from the instance I of MAX-3-LIN(2). Lets say a solution to an instance $R(I)$ of GAP is a k -assignment solution if exactly k switch items have been allocated. We will use *valid-assignment* to refer to the n -assignment solution.

Lemma 4.14 *For every solution of $R(I)$ which is a k -assignment solution such that variable x_i is unassigned (i.e. item s_i is neither allocated to $\langle x_i : 0 \rangle$ nor $\langle x_i : 1 \rangle$) there exists a k -assignment solution of at least the same value in which x_i is unassigned and both $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$ together gets atmost one item from S_e for every equation e of x_i . i.e. they both get a total of atmost $deg(x_i)$ items.*

Proof: Suppose not. i.e. there exists an equation e (say $x_i + x_j + x_k = \alpha$) s.t. both $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$ together gets atleast two items out of S_e . Notice that the switch items of x_j and x_k can fill the capacity of atmost one bin out of their respective two bins. Suppose the free bins are $\langle x_j : \alpha \rangle$ and $\langle x_k : \alpha \rangle$ (The other cases can be considered similarly). Now except for the item $\langle x_i : \alpha, x_j : \bar{\alpha}, x_k : \bar{\alpha} \rangle$, all the other 3 items in S_e are wanted by $\langle x_j : \alpha \rangle$ and $\langle x_k : \alpha \rangle$. By the above property, all of these 3 items can be allcated to the bins $\langle x_j : \alpha \rangle$ and $\langle x_k : \alpha \rangle$. Thus we can reallocate the items of S_e such that atmost one item out of S_e is allocated to the corresponding bins of variable x_i without decreasing the profit. \square

Now by the above lemma, for any unassigned variable x_i in a k -assignment solution, one of the bins out of $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$ will have atmost $deg(x_i)/2$ items. We can remove these items and allocate the switch element of x_i without reducing the profit. Thus we get the following corollary.

Corollary 4.15 *For every optimal solution of $R(I)$ which is a k -assignment solution there exists a $(k+1)$ -assignment solution which is also optimal. Therefore, there exists a optimal solution of $R(I)$ which is a valid assignment*

Now using arguments similar to lemma 4.5 , one can show the following:

Lemma 4.16 *Consider a valid-assignment solution (say v -sol) of $R(I)$. If an equation e is satisfied by the assignment of variables given by v -sol then all the 4 items of S_e can be allocated in v -sol. Otherwise atmost 3 items out of S_e can be allocated in v -sol.*

Now using arguments similar to lemma 4.6 and 4.7, one can prove the following lemma which along with Håstad's theorem implies Theorem 4.13.

Lemma 4.17 *Let I be an instance of MAX-3-LIN(2) and $R(I)$ be its reduction to GAP, then:*

- *If $OPT(I) \geq m(1 - \epsilon)$, then the maximum profit of $R(I)$ is at least $5.5m - 4m\epsilon$.*

- If $OPT(I) \leq m(1/2 + \eta)$, then the maximum profit of $R(I)$ is at most $5m + m\eta$

Proof: Suppose $OPT(I) \geq m(1 - \epsilon)$. Allocate switch elements in $R(I)$ so that the assignment of variables is same as the one given by optimal solution of I . Now the profit from switch items equals: $\sum_i (deg(x_i)/2) = 3m/2$. Also by lemma 4.16, the profit from *equation* items is atleast $4m(1 - \epsilon)$. Combining both, we get the first part of the lemma.

Suppose $OPT(I) \leq m(1/2 + \eta)$. By corollary 4.15, there exists a optimum solution of $R(I)$ which is a *valid* assignment. Consider any such solution. Now the claim is that for no more than $m(1/2 + \eta)$ equations, all the 4 items of S_e 's can be allocated in this solution. If they do, then along with lemma 4.16 it contradicts the fact that $OPT(I) \leq m(1/2 + \eta)$. Thus profit from equation items can be atmost: $4m(1/2 + \eta) + 3m(1/2 - \eta) = 7m/2 + m\eta$. Hence total profit can be atmost $3m/2 + 7m/2 + m\eta$. \square

4.4 Hardness of weighted MSSF

Given an undirected graph G , the unweighted maximum spanning star forest problem (MSSF) is to find a forest with as many edges such that each tree in the forest is a star. The edge-weighted MSSF (eMSSF) is the natural generalization with weights on edges. The node-weighted MSSF (nMSSF) 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.

It is clear that the non-trivial part of the above problem is to identify the vertices which are the centers of the stars. Once more, we reduce MAX-3-LIN(2) to both eMSSF and nMSSF. Let us discuss the edge-weighted MSSF first. For every variable x in an instance of MAX-3-LIN(2), we introduce two vertices: $\langle x : 0 \rangle$, $\langle x : 1 \rangle$. The interpretation is clear: we will enforce that exactly one of these vertices will be the center which will coincide with the assignment in the MAX-3-LIN(2) instance. Such an enforcing is brought about by putting a heavy cost edge between the vertices. For every equation we will add vertices as we did in the reduction to GAP.

In the node-weighted MSSF, we need to add an extra vertex, the *switch vertex*, along with the two vertices $\langle x : 0 \rangle$, $\langle x : 1 \rangle$. These vertices form a triangle and have a weight high enough to ensure that exactly one of $\langle x : 0 \rangle$, $\langle x : 1 \rangle$ is chosen as a center in any optimum solution to nMSSF.

We remark that Chen et.al [CEN⁺07] also use a similar gadget as the one above, although their reduction is from a variation of MAX-3-SAT and thus their results are weaker.

Hardness of eMSSF: Let I be an instance of MAX-3-LIN(2). Denote the variables as x_1, \dots, 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 $E(I)$ of eMSSF as follows:

- For every variable x_i , we have two variables which we label as $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$, corresponding to the two assignments. These variables are called variable vertices. There is an edge between them of weight $deg(x_i)/2$.
- For every equation $e : x_i + x_j + x_k = \alpha$ ($\alpha \in \{0, 1\}$), we have 4 vertices 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 $\langle x_i : \bar{\alpha}, x_j : \alpha, x_k : \bar{\alpha} \rangle$. This set of vertices are called equation vertices denoted by S_e . Thus we have $4m$ equation vertices in all. Each equation vertex of the form $\langle x_i : \alpha_i, x_j : \alpha_j, x_k : \alpha_k \rangle$ is connected to three variable vertices: $\langle x_i : \alpha_i \rangle$, $\langle x_j : \alpha_j \rangle$ and $\langle x_k : \alpha_k \rangle$. The weight of all these edges is 1. Thus, the degree of every equation vertex is 3 and the degree of every variable vertex $\langle x_i : 0 \rangle$ or $\langle x_i : 1 \rangle$ is $2deg(x_i)$.

Lemma 4.18 *Given any solution to $E(I)$, there exists a solution of at least the same weight where the centers are exactly one variable vertex per variable.*

Proof: Firstly note that if none of the variable vertices are centers then one can make one of them a center and connect the other to it and get a solution of higher cost (note that the degrees of the variables in the equations can be assumed to be bigger than 4 by replication). The proof is in two steps. Call a variable $x_i \in I$ *unassigned* if both of $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$ is a center. Call a solution of $E(I)$ k -satisfied if exactly k of the variables are assigned. The claim is that there exists a solution of equal or more weight which is n -satisfied. We do this via induction.

We show that if a solution to $E(I)$ is k -satisfied with $k < n$, then we can get a solution of at least this weight which is $k+1$ -satisfied. Pick a variable x_i which is unassigned. For every equation $e : x_i + x_j + x_k = 0$, say, containing x_i we claim that one can assume of the four equation vertices in S_e , only one is connected to $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$. This is because at least one of the two variable vertices corresponding to both x_j and x_k are centers. Suppose these are $\langle x_j : 0 \rangle$ and $\langle x_k : 0 \rangle$. Now note that of the four vertices in S_e only $\langle x_i : 0, x_j : 1, x_k : 1 \rangle$ is not neighboring to either of these centers. The three remaining can be moved to these without any decrease in the weight of the solution and the claim follows.

Thus, we can assume that for every unassigned variable x_i in the k -satisfied solution, one of the two variable vertices $\langle x_i : 0 \rangle$ or $\langle x_i : 1 \rangle$ (say $\langle x_i : 0 \rangle$), is connected to at most $\deg(x_i)/2$ equation vertices. Therefore, disconnecting all the equation items connected to $\langle x_i : 0 \rangle$, making it a leaf and connecting it to $\langle x_i : 1 \rangle$, gives a $k+1$ -satisfied solution of weight at least the original weight. \square

Now we get the hardness of eMSSF using the theorem of Håstad.

Theorem 4.19 *For any $\epsilon > 0$, it is NP-hard to approximate edge-weighted MSSF to a factor $10/11 + \epsilon$.*

Proof: The proof follows from the following two calculations and Theorem 4.2.

- If $OPT(I) \geq m(1 - \delta)$, then the maximum profit of $E(I)$ is at least $5.5m - 4m\delta$. For every variable x_i , if the assignment of x_i is $\alpha \in \{0, 1\}$, make $\langle x_i : \alpha \rangle$ the center. Observe that for every satisfied equation $e : x_i + x_j + x_k = \alpha$, all the four vertices of S_e can be connected to a center. Thus the weight of $E(I)$ is at least $\sum_i \deg(x_i)/2 + 4m(1 - \delta) = 5.5m - 4m\delta$.
- If $OPT(I) \leq m(1/2 + \eta)$, then the maximum profit of $E(I)$ is at most $5m + m\eta$. From the claim above we can assume for each variable x_i , one of its two variable vertices is a center. This defines an assignment of truth values to the variables and around half of the equations are not satisfied by this assignment. The observation is that for any unsatisfied equation $e : x_i + x_j + x_k = \alpha$, one of the four equation vertices in S_e is not connected to any center. Thus, the total weight of any solution is at most $\sum_i \deg(x_i)/2 + 4m(1/2 + \eta) + 3m(1/2 - \eta) = 5m + m\eta$.

\square

Hardness of nMSSF: Let I be an instance of MAX-3-LIN(2). The only difference between the instance of nMSSF, $N(I)$, and the eMSSF $E(I)$ is that for every variable $x_i \in I$, along with the variable vertices $\langle x_i : 0 \rangle$ and $\langle x_i : 1 \rangle$, we have a switch vertex s_i . The three vertices from a triangle and the node-weights of all of them are $\deg(x_i)/2$. The rest of the instance of $N(I)$ is exactly like $E(I)$ with the edge-weights being replaced by node-weights of 1 on the equation vertices.

The reason we require the third switch vertex per variable is that otherwise we cannot argue that in any solution to $N(I)$ at least one of the variable vertices should be a center. With the switch vertex, we can argue that this is the case. If none of the variable vertices is a center, then

the switch item is not connected to any vertex. Thus making any one of the variable vertices as a center connected to the switch item gives a solution to $N(I)$ of weight at least the original weight.

Lemma 4.18 now holds as in the case of $E(I)$ and thus similar to Theorem 4.20 we have the following hardness of node-weighted MSSF.

Theorem 4.20 *For any $\epsilon > 0$, it is NP-hard to approximate edge-weighted MSSF to a factor $13/14 + \epsilon$.*

Proof: The proof follows from the following two calculations and Theorem 4.2.

- If $OPT(I) \geq m(1 - \delta)$, then the maximum profit of $N(I)$ is at least $7m - 4m\delta$. For every variable x_i , if the assignment of x_i is $\alpha \in \{0, 1\}$, make $\langle x_i : \alpha \rangle$ the center. Connect the switch item and the vertex $\langle x_i : \bar{\alpha} \rangle$ to this center. Observe that for every satisfied equation $e : x_i + x_j + x_k = \alpha$, all the four vertices of S_e can be connected to a center. Thus the weight of $N(I)$ is at least $\sum_i deg(x_i) + 4m(1 - \delta) = 7m - 4m\delta$.
- If $OPT(I) \leq m(1/2 + \eta)$, then the maximum profit of $N(I)$ is at most $6.5m + m\eta$. From the claim above we can assume for each variable x_i , one of its two variable vertices is a center. This defines an assignment of truth values to the variables and around half of the equations are not satisfied by this assignment. The observation is that for any unsatisfied equation $e : x_i + x_j + x_k = \alpha$, one of the four equation vertices in S_e is not connected to any center. Thus, the total weight of any solution is at most $\sum_i deg(x_i) + 4m(1/2 + \eta) + 3m(1/2 - \eta) = 6.5m + m\eta$.

□

5 Discussion

In this chapter 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 of it: the integrality gap of the LP is $3/4$. An approach to get better approximation algorithms might be looking at stronger LP relaxations to the problem. One such relaxation is the *configurational LP relaxation* which we describe below.

5.1 The configurational LP relaxation

In this relaxation, we have a variable $x_{i,S}$ for every agent i and every subset of items $S \subseteq Q$. The LP is as follows

$$\begin{aligned}
 & \text{Max} && \left\{ \sum_i \sum_{S \subseteq Q} u_i(S) x_{i,S} \right. && (4) \\
 \text{s.t.:} & && \forall i \in A, && \sum_{S \subseteq Q} x_{i,S} \leq 1; \\
 & && \forall j \in Q, && \sum_{i, S \subseteq Q: j \in S} x_{i,S} \leq 1; \\
 & && \forall i \in A, S \subseteq Q, && x_{i,S} \geq 0 \}
 \end{aligned}$$

The first constraint implies that each agent gets at most one subset of items. The second implies that each item is at most one subset. The value generated on giving a subset S to agent i is

$$u_i(S) = \min(B_i, \sum_{j \in S} b_{ij}).$$

Solving the LP: Observe that the LP has exponentially (in n and m) many variables and an obvious question is how to solve the LP. One does so by going to the dual LP which will have exponentially many constraints but only polynomially many variables. Such a trick is now standard first used by Carr and Vempala [CV02]. The dual of LP 4 is as follows

$$\begin{aligned} \text{Min} \quad & \left\{ \sum_{i \in A} \alpha_i + \sum_{j \in Q} p_j \right. & (5) \\ \text{s.t.} \quad & \forall i \in A, S \subseteq Q, \quad \alpha_i + \sum_{j \in S} p_j \geq u_i(S); \\ & \forall i \in A, \forall j \in Q, \quad \alpha_i, p_j \geq 0 \end{aligned}$$

Suppose, for the time being, the LP5 has a separation oracle: Given (α_i, p_j) one can say in polynomial time if it is feasible for LP5 or find a subset of agents S with $\alpha_i + \sum_{j \in S} p_j < u_i(S)$. If so, then the ellipsoid algorithm can be used to solve the LP by making a polynomial number of queries to the separation oracle. Moreover, the subsets returned by the separation oracle are enough to describe the optimal solution to the dual. In other words, in the primal LP 4, only the variables corresponding to these constraints need be positive and the rest can be set to 0 and the optimum is not changed. Since they are only polynomially many, LP 4 can now be solved in polynomial time. Moreover, if one had an r -separation oracle ($r \leq 1$): Given (α_i, p_j) one can say in polynomial time if $\frac{1}{r} \cdot (\alpha_i, p_j)$ is feasible for LP5 or find a subset of agents S with $\alpha_i + \sum_{j \in S} p_j < u_i(S)$; then the above argument can be used to get an r -approximation for the primal LP 4.

Note that the separation problem for LP 5 is precisely the demand oracle problem described Section 1.2: given prices p_j to items, for every agent i find a subset of items maximizing $(u_i(S) - \sum_{j \in S} p_j)$ where $u_i(S) = \min(B_i, \sum_{j \in S} b_{ij})$. The problem is NP-hard with a reduction from PARTITION. Given an instance of partition of n integers b_1, b_2, \dots, b_n and a target integer B , consider the instance of the separation problem with n items having bids b_1 to b_n and prices $b_1/2, \dots, b_n/2$ and budget B . Note that the maximum value of the separation problem is at most $B/2$ and moreover the optimum is $B/2$ if and only if there is a subset of the integers adding exactly to B .

We now demonstrate an $(1 - \epsilon)$ -separation oracle for LP 5 using the FPTAS for the knapsack problem. The sketch below is not the fastest implementation as we interested mainly in the existence of polynomial time algorithms. In the knapsack problem, we are given a capacity of B and n items having profits p_j and weight w_j and the goal is to obtain a subset of items having maximum profit with the total weight being less than B . The problem is NP-hard, but an FPTAS exists. Moreover, there is an exact algorithm which runs in time $O(n^2P)$ where P is the largest profit. Given the separation problem for LP 5, we consider the items in any arbitrary order. The first item has a bid of b_1 and price p_1 . Suppose the item is picked in the optimum solution, call it S . If so, then $\sum_{j \in S} b_{ij} \leq B + b_1$, as otherwise one could discard the first item and get a better solution. Thus the optimum solution of the separation problem given the first item is picked is precisely

$$\max_{0 \leq x \leq b_1} \{ \text{Knapsack}[\{(b_2 - p_2, b_2), \dots, (b_n - p_n, b_n)\}, B - x] + (x - p_1) \}$$

$(x - p_1)$ is the value of the item 1 after the items from b_2, \dots, b_n use up $B - x$ of the budget. One can repeat the procedure n times removing one item at a time and in the end taking the best solution over all iterations. This gives the optimum solution in time $O(n^3B^2)$, where B is the budget.

To make the above algorithm run in polynomial time, we round down to the nearest integer all the budgets, bids and prices by a factor of $\epsilon B/n$. The solution to this reduced instance can be

found in time $O(n^5/\epsilon)$ and the same solution, scaled back, can be shown to be within $(1 - \epsilon)$ of the optimal solution (see for example, [Vaz02]).

Integrality gap of the configurational LP: In the next theorem we show that the integrality gap of the configurational LP is between $3/4$ and $5/6$. The lower bound follows basically by showing that the value of LP 4 is at most the value of LP 1 (and thus is a better upper bound on the optimum). The upper bound follows from an example which we demonstrate below.

Theorem 5.1 *The integrality gap of the configurational LP of MBA is between $3/4$ and $5/6$.*

Proof: An easy way to see that the configurational LP is stronger than LP(1) is by looking at the duals of both LP's. One can show that any solution (α_i, p_j) to LP(2) corresponds to a feasible solution $(B_i\alpha_i, p_j)$ to LP 5 of equal value. Thus the configurational LP value is smaller than that of LP(1).

The $5/6$ -example is as follows: The instance consists of 4 agents a_1, b_1, a_2, b_2 . a_1, a_2 have a budget of 1, b_1, b_2 have a budget of 2. There are five items: c, x_1, y_1 and x_2, y_2 . Only b_1 and b_2 bid on c and bid 2. For $i = 1, 2$, a_i and b_i each bid on x_i and y_i , and the bid is 1.

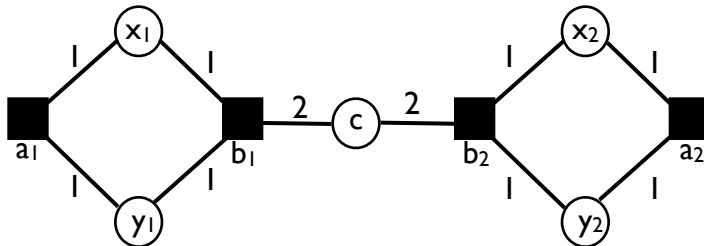


Figure 4: Integrality gap example for configurational LP.

Once again, if c is given to b_1 , then either a_2 or b_2 ends up spending 1 less than his budget. Thus, the optimum MBA solution is 5. But there is a solution to the configurational LP(4) of value 6. The sets are $S_1 = \{x_1\}$, $S_2 = \{y_1\}$, $S_3 = \{x_2\}$, $S_4 = \{y_2\}$, $S_5 = \{c\}$, $S_6 = \{x_1, y_1\}$ and $S_7 = \{x_2, y_2\}$. The solution is: $x_{a_1, S_1} = x_{a_1, S_2} = x_{a_2, S_3} = x_{a_2, S_4} = 1/2$ and $x_{b_1, S_5} = x_{b_1, S_6} = x_{b_2, S_5} = x_{b_2, S_7} = 1/2$. \square

The above theorem is about all we know for the configurational LP relaxation for MBA. We believe that the integrality gap should be strictly better (larger) than $3/4$ although it is not clear how to do so. Configurational LPs have been used for other allocation problems; in fact the best known approximation algorithm of Feige and Vondrák [FV06] for SMW and GAP proceeds by rounding the solution of the LP. However, we do not know how to use the “simplicity” of the submodular functions of MBA to get a better bound. (Feige and Vondrák get a factor strictly bigger than $1 - 1/e$). We leave the question of pinning down the exact integrality gap of this LP as an open question and believe the resolution might require some new techniques.

References

- [ABK⁺08] Y. Azar, B. Birnbaum, A. Karlin, C. Mathieu, and C. Nguyem. Improved approximation algorithms for budgeted allocations. *Proceedings of 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 186–197, 2008.

- [AM04] N. Andelman and Y. Mansour. Auctions with budget constraints. *Proceedings of 9th Scandinavian Workshop on Algorithm Theory SWAT*, pages 26–38, 2004.
- [And06] N. Andelman. *Online and strategic aspects of network resource management algorithms*. PhD thesis, Tel Aviv University, Tel Aviv, Israel, 2006.
- [BJN07] N. Buchbinder, K. Jain, and S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. *Proceedings of 15th Annual European Symposium ESA*, pages 253–264, 2007.
- [BK01] J-P. Benoit and V. Krishna. Multiple object auctions with budget constrained bidders. *Review of Economic Studies*, 68:155–179, 2001.
- [BN07] 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.
- [CC02] M. Chlebik and J. Chlebikova. Approximation hardness of the Steiner tree problem on graphs. *Proceedings of Scandinavian Workshop on Algorithm Theory*, pages 170–179, 2002.
- [CEN⁺07] N. Chen, R. Engelberg, C. Nguyen, P. Raghavendra, A. Rudra, and G. Singh. Improved approximation algorithms for the spanning star forest problem. *Proceedings of Approximation Algorithms for Combinatorial Optimization, International Workshop (APPROX)*, pages 44–58, 2007.
- [CK00] C. Chekuri and S. Khanna. A PTAS for the multiple-knapsack problem. *Proceedings of 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 213–222, 2000.
- [CV02] R. Carr and S. Vempala. Randomized metarounding. *Random Struct. and Algorithms*, 20:343–352, 2002.
- [FGMS06] L. Fleischer, M. X. Goemans, V. Mirrokini, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. *Proceedings of 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 611–620, 2006.
- [FV06] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. *Proceedings of 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 667–676, 2006.
- [GKP01] R. Garg, V. Kumar, and V. Pandit. Approximation algorithms for budget-constrained auctions. *Proceedings of Approximation Algorithms for Combinatorial Optimization, International Workshop (APPROX)*, pages 102–113, 2001.
- [Häs01] J. Hästad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.
- [KLMM05] S. Khot, R. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Proceedings of WINE*, pages 92–101, 2005.

- [LLN01] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Proceedings of 3rd ACM Conference on Electronic Commerce (EC)*, pages 18–28, 2001.
- [LST90] J. K. Lenstra, D. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46:259–271, 1990.
- [MSVV07] A. Mehta, A. Saberi, U. V. Vazirani, and V. V. Vazirani. Adwords and generalized on-line matching. *Journal of the ACM*, 54(5):1–22, 2007.
- [NSH⁺07] 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 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 645–654, 2007.
- [SS06] T. Sandholm and S. Suri. Side constraints and non-price attributes in markets. *Games and Economic Behavior*, 55(2):321–330, 2006.
- [ST93] D. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Math. Programming*, 62:461–474, 1993.
- [Vaz02] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2002.
- [Von08] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. *To appear in Proceedings of 40th Annual ACM Symposium on the Theory of Computing (STOC)*, 2008.