# Improved Approximation Algorithms for Budgeted Allocations*

Yossi Azar[1], Benjamin Birnbaum[2], Anna R. Karlin[2], Claire Mathieu[3], and C. Thach Nguyen[2]

[1] Microsoft Research and Tel-Aviv University
azar@tau.ac.il
[2] University of Washington
{birnbaum,karlin,ncthach}@cs.washington.edu
[3] Brown University
claire@cs.brown.edu

**Abstract.** We provide a 3/2-approximation algorithm for an offline budgeted allocations problem with applications to sponsored search auctions. This an improvement over the $e/(e-1)$ approximation of Andelman and Mansour [1] and the $e/(e-1)-\epsilon$ approximation (for $\epsilon \approx 0.0001$) of Feige and Vondrak [2] for the more general Maximum Submodular Welfare (SMW) problem. For a special case of our problem, we improve this ratio to $\sqrt{2}$. We also show that the problem is APX-hard.

## 1 Introduction

The rising economic importance of online sponsored search advertising has led to a great deal of research focused on developing its theoretical underpinnings. (See e.g., [3] for a survey.) Since search engines such as Google, Yahoo! and MSN depend on sponsored search for a significant fraction of their revenue, a key problem is how to optimally allocate ads to keywords (user searches) so as to maximize search engine revenue [1,4,5,6,7]. In this direction, Mehta et al. [7] studied a stylized version of the problem, which we call the *Online Budgeted Allocation* problem. In their model, there is a set of bidders $U$ and a set of keywords $V$. Each bidder $i \in U$ has a known daily budget $B_i$ and a non-negative bid $b_{ij}$ for every keyword $j \in V$. The keywords arrive one-by-one in an online fashion, with the bids for keyword $j$ revealed only when $j$ arrives. At each keyword arrival, the algorithm (i.e., the search engine) allocates the keyword to one of the bidders (i.e., displays that bidder's ad as one of the sponsored search results the user sees). The total profit extracted by the algorithm from each bidder is the minimum of the budget $B_i$ of that bidder and the sum of the $b_{ij}$'s for keywords $j$ allocated to it. The goal is to find an allocation of keywords to bidders that maximizes the total profit extracted by the algorithm. Mehta et al. [7] presented an algorithm that achieves an optimal competitive ratio of

* This research was supported by the Israeli Science Foundation, NSF Grant CCF-0635147, and by an NSF Graduate Research Fellowship.

L. Aceto et al. (Eds.): ICALP 2008, Part I, LNCS 5125, pp. 186–197, 2008.

$e/(e-1)$ for the case when the bids are much smaller than the budgets, a result also proved by Buchbinder et al. [4]. When there is no restriction on the values of the bids relative to the budgets, the best known competitive ratio is 2 [8].

Surprisingly, the approximability of the *offline* version of Budgeted Allocation, in which the algorithm can see all of the bids before allocating keywords, is still not well understood. Lehmann et al. [8] showed that the problem is NP-hard, and Andelman and Mansour [1] provided the first non-trivial approximation ratio of $e/(e-1)$. Feige and Vondrak [2] improved this ratio to $e/(e-1)-\epsilon$ (for $\epsilon \approx 0.0001$) for the more general SMW problem.

In this paper, we give improved approximation algorithms for two versions of the offline Budgeted Allocation problem. In the *uniform* version, each keyword $j$ has a single price $b_j$. If a bidder $i$ is interested in $j$, its bid $b_{ij}$ is equal to $b_j$. Otherwise, its bid $b_{ij}$ is 0. The *non-uniform* version removes this restriction, so that the $b_{ij}$ values can be arbitrary for each $i$ and $j$.

### 1.1 Our Results

We provide a deterministic 3/2-approximation algorithm for the non-uniform Budgeted Allocation problem (Section 2). This improves the previous best-known approximation ratio of $e/(e-1)-\epsilon$ (for $\epsilon \approx 0.0001$) [2]. For the uniform version of the problem, we improve the approximation ratio to $\sqrt{2}$ (Section 3). In both these algorithms, we assume that the maximum bid is no larger than the smallest budget, i.e. $\max_{i,j} b_{ij} \leq \min_i B_i$.

We also show that the problem is APX-hard (Section 4).

### 1.2 Related Work

As discussed above, before this work, the first non-trivial approximation ratio for Budgeted Allocation was $e/(e-1)$, due to Andelman and Mansour [1]. For the special case in which the bidders all have the same budget, these authors were able to lower this ratio to approximately 1.39. Our algorithms apply to the more general case in which the budgets may be different for different bidders.

Two recent unpublished works also study the Budgeted Allocation problem and provide better approximation ratios than those obtained in this paper: Independently of our work, Chakrabarty and Goel [9] have provided two elegant algorithms, an iterative rounding algorithm and a primal-dual algorithm, both of which achieve an approximation ratio of 4/3, matching the integrality gap of the linear program used in this and other papers. Their paper also shows that it is $NP$-hard to approximate Budgeted Allocation to a factor better than 16/15, which subsumes the result in this paper on APX-hardness. In addition, building on our approach, Srinivasan [10] has recently provided another LP-rounding algorithm that achieves an approximation ratio of 4/3.

The Budgeted Allocation problem is an important special case of SMW, the problem of maximizing utility in a combinatorial auction in which the utility functions are submodular. In the combinatorial auction setting the keywords are items to be sold. SMW has been widely studied [2,8,11,12,13,14]. For submodular

auctions using the value query model, the best approximation algorithm gives a factor $e/(e-1)$ [14]. This ratio has been shown to be the best possible for this model [12,13]. For the stronger demand query model it is possible to do at least slightly better, that is $e/(e-1)-\epsilon$ (for $\epsilon \approx 0.0001$) [2]. For solving the Budgeted Allocation problem using the SMW demand query model one needs to provide a polynomial-time demand query oracle. As noted by [15], one can use a knapsack-type FPTAS algorithm to provide an approximate oracle that is good enough for solving the problem.

The Budgeted Allocation problem is also similar to the *generalized assignment problem* (GAP) [2,15,16,17]. The main difference between Budgeted Allocation and GAP is that in GAP every keyword (or *item*, in GAP parlance) has a weight and each bidder (or *bin*) has a fixed capacity that cannot be exceeded, whereas in Budgeted Allocation, budgets can be exceeded, but no extra profit is obtained from doing so. The best approximation algorithm for GAP does slightly better than $e/(e-1)$ [2].

## 2 The 3/2-Approximation Algorithm

In this section, we describe the 3/2-approximation for the non-uniform version of the problem.

### 2.1 High-Level Idea

Our algorithms use linear program rounding. We represent the Budgeted Allocation problem with the same natural integer program used in [1,9,10], in which the 0-1 variables $x_{ij}$ represent whether keyword $j$ is allocated to bidder $i$:

$$\max \sum_{i \in U} \min(B_i, \sum_{j \in V} b_{ij} x_{ij}) \quad \text{s.t.} \quad \begin{cases} \sum_{i \in U} x_{ij} \leq 1 \ \forall j \in V \\ x_{ij} \in \{0,1\} \quad \forall i \in U, j \in V \end{cases} .$$

Let $L$ be the linear relaxation of this integer program in which the second constraint is replaced by $x \geq 0$. (The upper-bound of 1 is guaranteed by the other constraint.) Rounding the optimal solution carefully is what allows us to beat the factor of $e/(e-1)$ of Andelman and Mansour [1].

For any fractional allocation $\mathbf{x}$, define the graph $G$ *induced by* $\mathbf{x}$ to be the bipartite graph over $U \cup V$ with an edge $\{i,j\}$ for every $x_{ij} > 0$, with weight $w_{ij} = b_{ij}x_{ij}$.[1] Rounding $\mathbf{x}$ can be viewed as a transformation of $G$ into another graph in which the degree of every keyword is 1. Our algorithms do this iteratively, at each step modifying local structures so that the degree of at least one new keyword is reduced to 1. Some weight in the objective function will be lost at each step, but we use a charging argument to bound this loss by 1/3 of the original value of $\mathbf{x}$.

[1] Notice that in a fractional solution, we can assume without loss of generality that no bidder's budget is exceeded. Therefore, the value of $\mathbf{x}$ is equal to the sum of the edge weights in $G$.

The first observation for the proofs is that one can assume that the graph $G$ induced by a feasible fractional solution to $L$ has a special structure. This observation was made for optimal fractional solutions by [1], proved in [18], and used by [9,10]. We use a slightly stronger version that holds for any feasible solution. This will allow us to assume that this special structure holds after each rounding step, not just at the beginning. Say that bidder $i$ is *saturated* if $\min(B_i, \sum_{j \in V} b_{ij} x_{ij}) = B_i$ and is *unsaturated* otherwise.

**Lemma 1.** *Any feasible solution* $\mathbf{x}$ *of* $L$ *with induced graph* $G$ *can be transformed, in polynomial time, to another feasible solution* $\tilde{\mathbf{x}}$ *that has an induced graph that is a subgraph of* $G$ *and that is a forest with at most one unsaturated bidder per tree.*

*Proof.* The proof follows by standard arguments and is very similar to the proof in [18]. It can be found in the full version of this paper.[2]

For a graph $G$ induced by a fractional solution, say that a bidder is *active* if it has at least one neighbor of degree 2 or more in $G$, and is *inactive* otherwise. In our charging argument for the 3/2-algorithm, we charge to the weight allocated to bidders when they move from being active to inactive. (Since this happens at most once for each bidder, each unit of profit in the optimal fractional solution is charged at most once.)

We call the process of transforming a subgraph to reduce the degrees of the keywords *rounding* the subgraph. In general, this process will remove some of the edges and transfer some of the weight removed to the edges that remain, while respecting the constraints of the LP. For example, suppose that two keywords $j_1$ and $j_2$ are allocated fractionally to bidder $i$ and fractionally to some other bidders $i_1$ and $i_2$, as shown in Fig. 1. One way to round this part of the graph would be to remove the edges $\{i, j_1\}$ and $\{i_2, j_2\}$. Once this is done, $i$ has some unused budget that can be used to transfer an additional fraction of $j_2$ from $i_2$ to $i$. In general, transferring weight in this manner will be essential to obtaining our approximation ratio.

The main idea of our proof is that in every tree with active bidders, there is a small local structure involving only a constant number of nodes, such that if we round that structure so as to minimize the resulting loss in the objective function, the loss is at most 1/3 of the budget spent by the bidders that become inactive.

### 2.2 The Algorithm

Our 3/2-approximation algorithm is given below by Algorithm 1. At each iteration of the while loop, our algorithm looks for one of the *interesting nodes* (Definition 4) and rounds keywords in the associated structure. For each of the four types of interesting nodes, we describe a rounding subroutine used by the

[2] The full version of this paper is available at `http://www.cs.washington.edu/homes/birnbaum/budgetedallocation.pdf`.

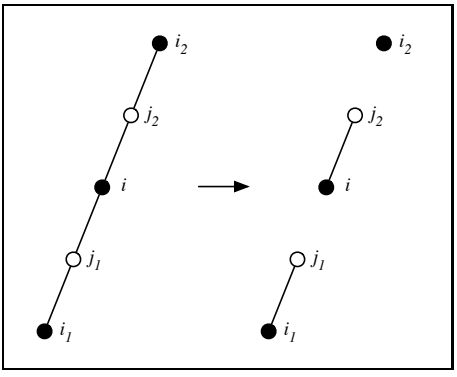


**Fig. 1.** One way to round a path that has three bidders. When edge $\{i, j_1\}$ is removed, this frees up some budget in $i$ to accommodate some of the weight of $j_2$ that was originally allocated to $i_2$.

algorithm. We will show that the loss of these subroutines can be charged to the nodes that become inactive, which we will use to prove that Algorithm 1 is a 3/2-approximation.

**Theorem 2.** *Algorithm 1 is a polynomial-time* 3/2*-approximation for the Budgeted Allocation problem.*

**Algorithm 1.** 3/2-approximation algorithm for Budgeted Allocation

**Input:** Set of bidders $U$, set of keywords $V$; for each $i, j$, bid $b_{ij}$ of bidder $i$ for keyword $j$; and for each bidder $i$, budget $B_i$.
**Output:** Allocation of keywords to bidders.
**solve** the following LP to get an optimal solution $\mathbf{x}$ with induced graph $G$:

$$\max \sum_{i \in U} \min(B_i, \sum_{j \in V} b_{ij} x_{ij}) \quad \text{s.t.} \quad \begin{cases} \sum_{i \in U} x_{ij} \leq 1 \ \forall j \in V \\ 0 \leq x_{ij} \qquad \forall i \in U, j \in V \end{cases} .$$

**transform** $G$ into a forest with $\leq 1$ unsaturated bidder per tree (Lemma 1).
**while** $G$ *contains active bidders* **do**
  Round the subgraph associated to an interesting node according to its type;
  Transform $G$ into a forest with $\leq 1$ unsaturated bidder per tree (Lemma 1);
**end**
**allocate** each keyword to its unique adjacent bidder in $G$.

Root each tree of $G$ at the unsaturated bidder if there is one, or at an arbitrary bidder if there is not.

**Definition 3.** *Consider a path* $i_1, j_1, i_2, j_2, \ldots, i_{k-1}, j_{k-1}, i_k$ *consisting of* $2k-1$ *nodes, starting and ending with a bidder, such that*

– *the* $k-1$ *keywords* $j_1, j_2, \ldots, j_{k-1}$ *all have degree exactly 2,*

- *bidder $i_1$ is the highest node on the path, called "root" of the path,*
- *for all other bidders $i_2, i_3, \ldots, i_k$, any keyword not on the path that is adjacent to the bidder has degree exactly 1.*

*We call the graph formed by this path and all the degree-1 keywords that are neighbors of $i_1, i_2, \ldots, i_k$ a* $k$-chain.

**Definition 4.** *In a tree of $G$, we say that a node is* interesting *if it has one of the following four types.*

1. *The root of the tree, if the tree consists of a 2-chain (Fig. 2(a)).*
2. *A bidder $v$ whose subtree contains at least one 3-chain rooted at $v$ (Fig. 2(b)).*
3. *A bidder $v$ who is the root of more than one 2-chain and who is not the root of a $k$-chain, for $k > 2$ (Fig. 2(c)).*
4. *A keyword with at least 2 children, such that each child is a root of a 1-chain or a 2-chain (Fig. 2(d)).*

Before we describe the rounding subroutines, we establish the correctness of Algorithm 1.

**Lemma 5.** *A tree that has a keyword of degree more than 1 must have at least one interesting node.*

*Proof.* A straightforward proof can be found in the full version of this paper.

Hence, in the forest produced by Algorithm 1, every keyword has degree 1, and the output is an integer allocation.

This lemma, along with the analysis of the rounding subroutines described below, will give us all of the ingredients we need to prove that Algorithm 1 is a 3/2-approximation.

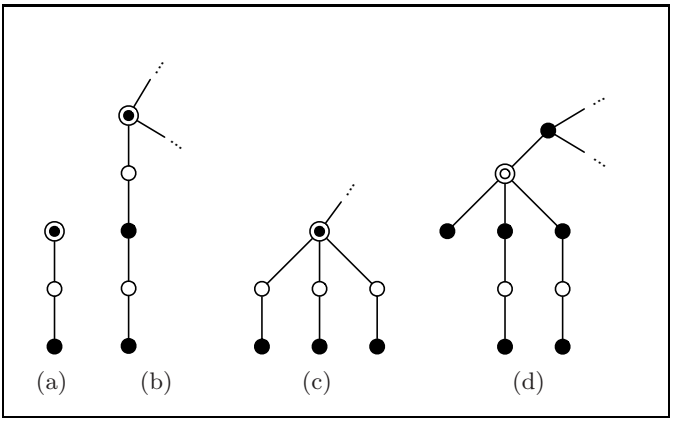


**Fig. 2.** Examples of the four types of interesting nodes with their associated subgraphs, as defined in Definition 4. For each type, the interesting node is shown with an extra circle.

*Proof of Theorem 2.* Lemma 5 proves that when Algorithm 1 terminates, it returns a graph in which each keyword has degree 1. Each rounding step described below clearly takes polynomial time and makes at least one new bidder become inactive. Hence, the running time of the algorithm is polynomial.

For each rounding, we will associate each unit lost to 1/3 of the weight spent by a bidder that becomes inactive. Since each bidder becomes inactive only once, the total weight lost must be no larger than 1/3 of the weight of the optimal fractional solution. Therefore, the algorithm returns a solution with weight at least 2/3 of the optimal fractional solution and hence with weight at least two thirds of the optimal integral solution. □

### 2.3 The Rounding Subroutines

To simplify the exposition, we assume, without loss of generality, that all of the bids and budgets have been scaled so that the maximum bid is 1 (and hence the minimum budget is at least 1).

#### Type 1 Rounding

*Rounding.* Let $i$ be the interesting node, $j$ be its child of degree 2, and $k$ be its grandchild. By Lemma 1, bidder $k$ is saturated, while bidder $i$ may have some unused budget $s \geq 0$.

Consider two ways to round the 2-chain rooted at $i$. In the first way, we remove the edge $\{i, j\}$. In the second way, we remove the edge $\{k, j\}$ and transfer as much as possible of the removed weight to the edge $\{i, j\}$ while maintaining feasibility. Of those two ways, we choose the one that incurs the smaller loss in the objective function.

*Analysis.* Since this rounding makes both $i$ and $k$ inactive, we can charge their total value, which is at least $\max(1, 2 - s)$. The following lemma states the performance of this rounding.

**Lemma 6.** *Let $L$ be the total weight lost by the rounding. Then $L \leq \frac{1}{4}\max(1, 2 - s)$.*

*Proof.* The proof is technical and can be found in the full version of this paper.

#### Type 2 and Type 3 Roundings

*Rounding.* In type 2 rounding, we have a path with two keywords of degree 2; we consider all four possible ways of allocating each of those two keywords integrally to one of its two neighbors, transferring as much weight as possible in each allocation while respecting the LP constraints.

In type 3 rounding, we take a partial subtree rooted at the interesting node consisting of two of the paths below it. Together, these two paths define a path with two keywords of degree 2. We then proceed as in type 2 to define an integer allocation of those two keywords.

*Analysis.* In all cases, two saturated bidders become inactive. We show that the loss in the objective function is no more than 2/3 (Lemma 7), and charge it to the total value of the two bidders that become inactive, which is at least 2.

More precisely, consider a path $i_1, j_1, i_2, j_2, i_3$ where $j_1$ and $j_2$ are keywords. We show four ways to round this path so that one of the edges $\{i_1, j_1\}$, $\{i_2, j_1\}$ and one of the edges $\{i_2, j_2\}$, $\{i_3, j_2\}$ is removed in a way that loses no more than 2/3 (Lemma 7). If this path is a 3-chain rooted at $i_1$, then this procedure makes $i_2$ and $i_3$ inactive. Hence, we can charge the loss to the total value of $i_2$ and $i_3$, which is at least 2. On the other hand, if $i_2$ is the highest node on this path, $j_1$ and $j_2$ are degree-2 keywords and $i_1$ and $i_3$ do not have any degree-2 children, then this procedure makes $i_1$ and $i_3$ inactive. Hence, we can charge the loss to the total value of $i_1$ and $i_3$, which is at least 2.

Let $\gamma = b_{i_2 j_1}/b_{i_1 j_1}$, $\beta = b_{i_2 j_2}/b_{i_3 j_2}$, $a = x_{i_1 j_1} b_{i_1 j_1}$, $b = x_{i_2 j_1} b_{i_1 j_1}$, $c = x_{i_2 j_2} b_{i_3 j_2}$ and $d = x_{i_3 j_2} b_{i_3 j_2}$. Then $w_{i_1 j_1} = a$, $w_{i_2 j_1} = b\gamma$, $w_{i_2 j_2} = c\beta$ and $w_{i_3 j_2} = d$. This situation is illustrated in Fig. 3(a).

We consider four ways to round the path, illustrated in Figs. 3(b)-3(e). In the first way (Fig. 3(b)), we remove the edges $\{i_1, j_1\}$ and $\{i_3, j_2\}$, losing $a + d$. In the second way (Fig. 3(c)), we remove the edges $\{i_2, j_1\}$ and $\{i_2, j_2\}$, losing $b\gamma + c\beta$. In the third way (Fig. 3(d)), we remove the edges $\{i_1, j_1\}$ and $\{i_2, j_2\}$ and move part of the removed weight to $\{i_2, j_1\}$. If the entire amount of $j_1$ that was previously allocated to $i_1$ were allocated to $i_2$, then $w_{i_2 j_1}$ would increase by $x_{i_1 j_1} b_{i_2 j_1} = (a/b_{i_1 j_1}) b_{i_2 j_1} = a\gamma$. The budget freed up at $i_2$ from the removal of edge $\{i_2, j_2\}$ is $c\beta$. Thus, $w_{i_2 j_1}$ can be increased to at least $b\gamma + \min(a\gamma, c\beta)$, causing a loss of $a + c\beta - \min(a\gamma, c\beta) = a + \max(0, c\beta - a\gamma)$. In the fourth way (Fig. 3(e)), we remove the edges $\{i_2, j_1\}$ and $\{i_3, j_2\}$ and transfer as much as possible of the removed weight to $\{i_2, j_2\}$, causing a loss of $d + \max(0, b\gamma - d\beta)$.

Again, we choose the way that incurs the smallest loss. The following lemma states that this loss is never greater than 2/3.

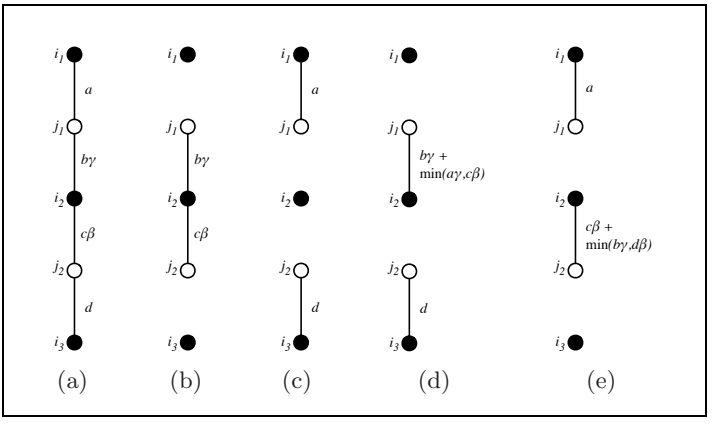


**Fig. 3.** A path with three bidders (a) and four ways to round that path (b)-(e)

**Lemma 7.** *Let*

$$L = \min\left(a + d, b\gamma + c\beta, a + \max(0, c\beta - a\gamma), d + \max(0, b\gamma - d\beta)\right) \quad .$$

*Then* $L \leq \frac{2}{3}$.

*Proof.* The proof is technical and can be found in the full version of this paper.

**Type 4 Rounding**

Let $v$ be the interesting node of type 4, and let $u$ be its parent. Let $h$ and $k$ be the number of 1-chains and 2-chains rooted at children of $v$, respectively. We consider three cases based on the value of $h$ and $k$:

1. There are no 2-chains attached to $v$ ($k = 0$). Then $h > 1$.
   *Rounding.* Among the edges adjacent to $v$, retain the edge of largest weight and delete all others.
   *Analysis.* We lose at most $h/(h+1)$ and make at least $h$ saturated bidders inactive. Therefore, we can charge the loss to these nodes.
2. There are no 1-chains attached to $v$ ($h = 0$). Then $k > 1$. Let $p_1, p_2, \ldots p_k$ be $v$'s children.
   *Rounding.* We first round the path consisting of the edges $\{u, v\}$, $\{v, p_1\}$ and the 2-chain rooted at $p_1$, losing at most 2/3 by Lemma 7. After this step, either $p_1$ or $u$ is disconnected from $v$. We repeat the above step with the path containing the edge joining $v$ and the other node (either $u$ or $p_1$), $\{v, p_2\}$ and the 2-chain rooted at $p_2$. We repeat this $k$ times.
   *Analysis.* We lose at most $2k/3$ and make $2k$ saturated bidders inactive: $p_1, p_2, \ldots p_k$ and their grandchildren. Hence, we can charge the loss to these nodes.
3. Both $h > 0$ and $k > 0$.
   *Rounding.* We choose one 1-chain rooted at, say, $q$, and one 2-chain rooted at, say, $p$ and round the path containing $q, v, p$ and the 2-chain rooted at $p$.
   *Analysis.* We lose at most 2/3 by Lemma 7 and make two saturated bidders inactive: $p$'s grandchild and either $q$ or $p$. Hence, we can charge the loss to these nodes.

## 3 A $\sqrt{2}$-Approximation Algorithm for the Uniform Problem

In this section, we provide an algorithm that improves the approximation ratio to $\sqrt{2}$ for the uniform case of the problem. The main observation that leads to this improvement is that in the proof of Theorem 2, there was some weight that we could have charged to but that we did not use. For example, consider the type 2 rounding shown in Figure 3(b). We charged the loss of the rounding to the weight allocated to bidders $i_2$ and $i_3$, which must be at least 2 since these bidders are saturated. We can do better than this, however. In the rounding, a weight of

$a$ is deallocated from $i_1$. Because we rebalance according to Lemma 1 between every rounding, this is weight that will never be charged to again. Therefore, instead of charging to 2, we can actually charge to $2 + a$.

To make this more precise, we define an *active edge* to be an edge that is adjacent to an active bidder. During each rounding, the sum of the weights on active edges will decrease, both from active edges becoming inactive and from active edges being deleted or losing weight. Define the *accountable amount* of a rounding to be the amount by which this quantity decreases. We will show that the loss of each rounding can be charged to $(1 - 1/\sqrt{2})$ of the accountable amount of that rounding. Since each unit of accountable amount is charged at most once, this suffices to prove the approximation ratio.

The structure of Algorithm 2, our $\sqrt{2}$-approximation, is the same as that of Algorithm 1. The only difference is in the rounding subroutines and their analysis. Instead of choosing the rounding that minimizes the loss at each step, we choose the one that minimizes the ratio between the loss and the accountable amount. In the remainder of this section we prove the following.

**Theorem 8.** *Algorithm 2 is a polynomial-time $\sqrt{2}$-approximation for the uniform version of the Budgeted Allocation problem.*

*Proof.* As in Theorem 2, the algorithm terminates in polynomial time and outputs an integral solution. For each rounding subroutine, we show that we can charge the loss to $1 - 1/\sqrt{2}$ of the accountable amount. This implies that Algorithm 2 returns a solution of value at least $1/\sqrt{2}$ of optimal. □

We believe that Theorem 8 applies to the non-uniform version of the problem, but we have not been able to prove this, since it seems to involve calculations that are significantly more complicated than those in the proof of Lemma 7.

### 3.1 The Rounding Subroutines

For convenience, we define $x$ to be $2 - \sqrt{2}$. For each rounding subroutine, we show that the ratio of the loss to the accountable amount is no greater than $x/2 = 1 - 1/\sqrt{2}$.

#### Type 1 Rounding

The rounding and analysis for type 1 is the same as for Algorithm 1. By Lemma 6, the ratio of the loss to the accountable amount is no greater than $1/4 < x/2$.

#### Type 2 Rounding

*Rounding.* Let $i_1$ be the interesting node and $i_1, j_1, i_2, j_2, i_3$ be the 3-chain associated with $i_1$, and let $a = w_{i_1 j_1}$, $b = w_{i_2 j_1}$, $c = w_{i_2 j_2}$ and $d = w_{i_3 j_2}$. Of the four ways to round this chain described in Fig. 3, we choose the rounding that minimizes the ratio of the loss over the accountable amount.

*Analysis.* The four ways to round the chain incur losses of $a+d$, $b+c$, $\max(a,c)$, and $\max(b,d)$, respectively. (Recall that in the uniform version $\beta = \gamma = 1$.) To derive the accountable amounts of the roundings, note that the first rounding makes $i_2$ and $i_3$ inactive, and thus makes all edges adjacent to these nodes inactive; the sum of these edges is at least 2, since these bidders are saturated. Furthermore, it also removes the active edge $\{i_1, j_1\}$. Thus, the accountable amount of the first rounding is $2 + a$. Similarly, the accountable amount of the second, third and fourth roundings are $2, 2 + a$ and 2, respectively. Hence, the following lemma shows that the we can always choose a rounding such that the ratio of the loss to the accountable amount is no greater than $x/2$.

**Lemma 9.** *Let*

$$R = \min\left(\frac{a+d}{2+a}, \frac{b+c}{2}, \frac{\max(a,c)}{2+a}, \frac{\max(b,d)}{2}\right) .$$

*Then* $R \leq x/2$.

*Proof.* The proof is technical and can be found in the full version of this paper.

### Type 3 and Type 4 Roundings

The rounding subroutine and analysis for type 3 interesting nodes is similar to the rounding and analysis for type 2 interesting nodes. The rounding and analysis for type 4 interesting nodes is quite involved, though the main ideas are similar. The details of type 3 and type 4 rounding can be found in the full version of this paper.

## 4 APX-Hardness

In this section, we show the following result.

**Theorem 10.** *Budgeted Allocation is APX-hard, even in the uniform version.*

*Proof.* The proof is a simple reduction from 3D-Matching, and can be found in the full version of this paper.

## References

1. Andelman, N., Mansour, Y.: Auctions with Budget Constraints. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111. Springer, Heidelberg (2004)
2. Feige, U., Vondrak, J.: Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In: FOCS 2006, pp. 667–676 (2006)
3. Lahaie, S., Pennock, D., Saberi, A., Vohra, R.: Sponsored search auctions. In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.) Algorithmic Game Theory, pp. 699–716. Cambridge University Press, Cambridge (2007)
4. Buchbinder, N., Jain, K., Naor, J.: Online primal-dual algorithms for maximizing ad-auctions revenue. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 253–264. Springer, Heidelberg (2007)

5. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to adwords. In: SODA 2008, pp. 982–991 (2008)
6. Mahdian, M., Nazerzadeh, H., Saberi, A.: Allocating online advertisement space with unreliable estimates. In: EC 2007, pp. 288–294 (2007)
7. Mehta, A., Saberi, A., Vazirani, U., Vazirani, V.: Adwords and generalized online matching. J. ACM 54(5), 22 (2007)
8. Lehmann, B., Lehmann, D., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. Games and Economic Behavior 55(2), 270–296 (2006)
9. Chakrabarty, D., Goel, G.: On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP (manuscript, 2008)
10. Srinivasan, A.: Budgeted allocations in the full-information setting (manuscript, 2008)
11. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: SODA 2006, pp. 1064–1073 (2006)
12. Khot, S., Lipton, R., Markakis, E., Mehta, A.: Inapproximability Results for Combinatorial Auctions with Submodular Utility Functions. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 92–101. Springer, Heidelberg (2005)
13. Mirrokni, V., Schapira, M., Vondrak, J.: Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions (manuscript, 2007)
14. Vondrak, J.: Optimal approximation for the submodular welfare problem in the value oracle model. In: STOC 2008 (to appear, 2008)
15. Fleischer, L., Goemans, M., Mirrokni, V., Sviridenko, M.: Tight approximation algorithms for maximum general assignment problems. In: SODA 2006, pp. 611–620 (2006)
16. Chekuri, C., Khanna, S.: A PTAS for the multiple knapsack problem. In: SODA 2000, pp. 213–222 (2000)
17. Shmoys, D., Tardos, E.: An approximation algorithm for the generalized assignment problem. Mathematical Programming 62, 461–474 (1993)
18. Andelman, N.: Online and strategic aspects of network resource management algorithms. PhD thesis, Tel Aviv University (2006)