Journal of the Operations Research Society of Japan
Vol. 64, No. 1, January 2021, pp. 31–44

APPROXIMATION ALGORITHMS FOR A GENERALIZATION OF THE MAXIMUM BUDGET ALLOCATION

Takuro Fukunaga
Chuo University

(Received September 4, 2019; Revised August 3, 2020)

Abstract The maximum budget allocation (MBA) problem is the problem of allocating items to agents so as to maximize the total payment from all agents, where the payment from an agent is the sum of prices of the items allocated to that agent, capped by the agent's budget. In this study, we consider a generalization of the MBA problem in which each item has a capacity constraint, and present two approximation algorithms for it. The first is a polynomial bicriteria algorithm that is guaranteed to output an allocation producing at least $1 - r$ times the optimal feasible total payment, where r is the maximum ratio of price to budget, and to violate the capacity constraints on items by at most a factor of 2. The other is a pseudo-polynomial algorithm with approximation ratio $1/3 \cdot (1 - r/4)$ that always outputs a feasible allocation.

Keywords: Combinatorial optimization, budget allocation problem, knapsack problem, LP-rounding algorithm

1. Introduction

The *maximum budget allocation (MBA) problem* is a problem of finding an optimal allocation of items to agents. In this problem, we are given a set of items and a set of agents. Each agent has a budget and bids for items. If agents are allocated items, then in principle they have to pay their bids for those items. However, each agent's payment is capped by their budget. The objective of the problem is to find an allocation that maximizes the total payment from the agents. This problem models many practical situations, particularly in advertising, and hence it has been extensively studied in various fields of computer science.

In the present paper, we consider a generalization of the MBA problem in which each item has a capacity constraint. We assume that each agent has a length and each item has a capacity. Unlike in the standard MBA problem, in which each item is allocated to at most one agent, in our generalized problem items can be allocated to more than one agent subject to the capacity constraint that the sum of agents' lengths does not exceed the capacity of the item. We call this problem the *maximum budget allocation problem under knapsack constraints* (knapsack MBA problem).

The knapsack MBA problem was introduced by Sumita et al. [15], who investigated online algorithms for the problem. Their motivation was to optimize video ad allocations. In their setting, the agents and items correspond to advertisers and viewers available to show video ads, and agent lengths and item capacities are respectively the lengths of the video ads and the total amounts of time that the viewers spend watching video ads. Solving the knapsack MBA problem means finding an optimal allocation of video ads to viewers. The knapsack MBA problem models many practical situations besides video ads. For example, even in traditional print advertising using words and still pictures, a reader looks at several ads, spending a certain amount of time with each, and hence a capacity constraint is still

meaningful. Therefore, investigating algorithms for this problem is worthwhile.

The aim of this paper is to present approximation algorithms for the offline version of the knapsack MBA problem. Prior to the present work, two approximation algorithms for the knapsack MBA problem have been reported:

- An algorithm with approximation ratio $\max_{0\leq x\leq 1/2}(1-2x)(1-1/e^x) \approx 0.11$. This algorithm combines the continuous greedy algorithm [3] and the monotone contention resolution scheme [6] for 1-sparse packing constraints.
- An algorithm with approximation ratio $(1-r)(1-1/(1+r)^{1/r})$ proposed by Sumita et al. [15], where r is the maximum ratio of bid to budget. As r approaches 0, the approximation ratio approaches $1-1/e$, whereas $r=1$ gives a ratio of 0.

The knapsack MBA problem is actually a special case of the monotone submodular maximization problem subject to 1-sparse packing constraints, which will be discussed in more detail in Section 2.1. Hence algorithms for the latter problem can be applied to the knapsack MBA problem. To our knowledge, the current best approximation ratio for the latter problem is 0.11 [3, 6]. The algorithm proposed by Sumita et al. [15] is an online algorithm but can also be applied to the offline problem. In their paper, they discuss the problem mainly under the *small bids assumption*, which assumes that r is close to 0. The small bids assumption is reasonable in the context of Internet advertising, where it is cheap to buy a single display of an ad, and many previous studies in this field have also adopted this assumption; see Section 2.2. However, there are also many situations in which the small bids assumption is not reasonable; even in the context of advertising, the expense of putting an ad on TV or in newspapers is sufficient for r to be large. The algorithm of Sumita et al. is good when the small bids assumption holds, but its approximation ratio is close to 0 when r is large. On the other hand, the algorithm for the submodular maximization problem achieves a constant approximation ratio regardless of r, but that ratio is low.

The contributions of this paper are the following two approximation algorithms:

- A polynomial bicriteria $(1-r, 2)$-approximation algorithm. The allocation output is guaranteed to produce at least $1-r$ times the optimal feasible total payment and to violate the capacity constraints by at most a factor of 2.
- A pseudo-polynomial $1/3 \cdot (1-r/4)$-approximation algorithm. Note that since $r \leq 1$, the approximation ratio is at least $1/4$.

In the first algorithm, the approximation ratio is better than that of the algorithm of Sumita et al. As r approaches 0, the ratio approaches 1 (i.e., revenue approaches the optimal value). However, the ratio is still close to 0 if r is close to 1. In addition, the algorithm is allowed to violate the capacity constraints. On the other hand, the second algorithm achieves at least a $1/4$ ratio for any r, which is better than that using the submodular maximization algorithm irrespectively of r, and is also better than those using the algorithm of Sumita et al. and the bicriteria algorithm when r is large. However, our second algorithm has the disadvantage that it is not polynomial.

Both of the proposed algorithms depend on linear programming (LP) relaxations of the problem. A key point for these algorithms is how the capacity constraints on items are handled. In the first algorithm, the LP relaxation represents each capacity constraint as a single inequality. We will show the properties of the extreme point solutions to the relaxation and present an iterative rounding algorithm. In the second algorithm, we use a time-indexed LP that represents the capacity constraint on an item as the time limit for processing all agents to which the item is allocated. We will show that a scaled solution for this time-indexed LP can be described as a convex combination of characteristic vectors of

feasible solutions. By outputting the best solution in the combination, we can achieve the required approximation ratio.

The remainder of this paper is organized as follows. Section 2 gives a formal statement of the problem, and discusses related previous studies. Section 3 presents our bicriteria $(1-r,2)$-approximation algorithm. Section 4 presents our $1/3 \cdot (1-r/4)$-approximation algorithm. Section 5 concludes the paper.

2. Preliminaries

2.1. Problem definition

The knapsack MBA problem is formally defined as follows. Let $\mathbb{R}_+$ denote the set of non-negative reals. For any positive integer k, we let $[k]$ and $[k]^+$ denote the sets $\{1,\dots,k\}$ and $\{0,\dots,k\}$, respectively. Let $N=[n]$ be a set of n agents, where each agent $i \in N$ has budget $B_i \in \mathbb{R}_+$ and length $l_i \in \mathbb{R}_+$. Let $M=[m]$ be a set of m items, where each item j has capacity $L_j \in \mathbb{R}_+$. Each agent i submits bid $b_{ij} \in \mathbb{R}_+$ for item j. The task in the knapsack MBA problem is to decide for each item j the set S_j of agents allocated that item. For each item j, the allocation is required to satisfy the capacity constraint that the total length of agents in S_j does not exceed the capacity of item j, i.e., $\sum_{i\in S_j} l_i \le L_j$ for each $j \in M$.

If agent i is allocated item j, then agent i pays b_{ij} from their budget. However, if the total payment exceeds the budget, the agent pays only what remains of their budget. In other words, when the allocation of item j is represented by $S_j \subseteq N$, the payment of agent i is $\min\{\sum_{j\in M: i\in S_j} b_{ij}, B_i\}$. The objective of the problem is to maximize the revenue, which is the total payment from the agents, $\sum_{i\in N} \min\{\sum_{j\in M:i\in S_j} b_{ij}, B_i\}$.

Throughout this paper, we let OPT denote the maximum revenue attained by feasible allocations. We define r as $\max_{i\in N, j\in M} b_{ij}/B_i$. We can assume without loss of generality that $b_{ij} \le B_i$ for any $i \in N$ and $j \in M$, and hence $0 \le r \le 1$.

2.2. Related problems

The knapsack MBA problem coincides with the MBA problem when $l_i = 1$ for all $i \in N$ and $L_j = 1$ for all $j \in M$. The online version of the MBA problem is known as the *adwords problem* [11] or *ad-auction problem* [2]. Here, we restrict our attention to the offline problem. Chakrabarty and Goel [4] showed that it is NP-hard to approximate the MBA problem within a ratio of less than $1-r/16$. Simultaneously, Srinivasan [14] and Chakrabarty and Goel [4] presented a $(1-r/4)$-approximation algorithm for the problem, which is to say an algorithm that always computes a feasible allocation giving revenue of at least $(1-r/4)\cdot \mathrm{OPT}$. Since $r \le 1$, this approximation ratio is at least $3/4$. This ratio has since been slightly improved by Kalaitzis [8].

When $r=1$ and there exists w_i $(i \in M)$ such that $b_{ij} \in \{w_i, 0\}$ for all $i \in N$ and $j \in M$, the knapsack MBA problem is equivalent to the problem of assigning each agent to one of a set of given knapsacks for which the agent bid a fixed positive weight. This problem is known as the multiple knapsack problem with assignment restrictions. Dawande et al. [7] gave a $1/2$-approximation algorithm for this problem. They also gave a bicriteria approximation algorithm, although their definition of bicriteria approximation guarantees differ from ours. In addition, if $r=1$ and $b_{ij}=w_i$ for all $i \in N$ and $j \in M$, then the knapsack MBA problem is equivalent to the multiple knapsack problem (without assignment restrictions). A polynomial-time approximation scheme is known for this problem [5].

The knapsack MBA problem can be regarded as a special case of the monotone submodular function maximization subject to 1-sparse packing constraints. Let $S=\{(i,j)\in N\times M\}$,

and define a function $f\colon 2^S \to \mathbb{R}_+$ by $f(S') = \sum_{i\in N} \min\{\sum_{j\in M:(i,j)\in S'} b_{ij}, B_i\}$ for $S' \subseteq S$. Then f is monotone and submodular. The knapsack MBA problem is equivalent to maximizing $f(S')$ subject to the knapsack constraint that $\sum_{i\in N:(i,j)\in S'} l_i \leq L_j$ for each $j \in M$. Note that each $(i,j) \in S$ appears in exactly one of the $|M|$ knapsack constraints. This setting is called being under 1-sparse packing constraints. For the maximization problem of a monotone submodular function subject to 1-sparse packing constraints, we can obtain a $(1-2x)(1-1/e^x)$-approximation algorithm for any $x \in [0, 1/2]$ by using the continuous greedy algorithm [3] and the monotone $(x, 1-2x)$-balanced contention resolution scheme [6]. The best approximation ratio achieved by this approach is $\max_{0\leq x\leq 1/2}(1-2x)(1-1/e^x) = 2(1/W(e^{1.5}) + W(e^{1.5}) - 2) \approx 0.11$, where W is the product logarithm, and this maximum is attained for $x = 1.5 - W(e^{1.5})$.

3. Polynomial Bicriteria $(1-r, 2)$-Approximation Algorithm

We propose a bicriteria approximation algorithm by relaxing the capacity constraint for each item. That is to say, we permit violations of the capacity constraints but simultaneously guarantee that the violations are bounded by some factor. We say that an algorithm is an (α, β)-*approximation* for $\alpha \leq 1$ and $\beta \geq 1$ if an allocation computed by the algorithm satisfies the following two guarantees for any feasible instance:

- revenue of the allocation is at least αOPT;
- for each item $j \in M$, the total length of agents allocated the item (i.e., $\sum_{i\in S_j} l_j$) is at most βL_j.

If $\beta > 1$, then the (α, β)-approximation algorithm does not always compute a feasible allocation. However, the second guarantee indicates that the capacity constraint is violated by a factor of at most β for each item.

We propose a $(1-r, 2)$-approximation algorithm. The guarantee on the revenue for our algorithm approaches 1 as r approaches 0. Hence, if each bid b_{ij} is sufficiently small relative to budget B_i, then our algorithm achieves nearly optimal revenue (although the capacity constraints are violated, but by at most a factor of two).

Our algorithm is as follows. Let $E = \{ij \mid i \in N, j \in M, l_i \leq L_j\}$. We regard E as a set of undirected edges on the vertex set $N \cup M$. The problem is equivalent to finding $F \subseteq E$ such that $\sum_{i\in N:ij\in F} l_i \leq L_j$ for each $j \in M$ and $\sum_{i\in N} \min\{\sum_{j\in M:ij\in F} b_{ij}, B_i\}$ is maximized. The following LP relaxes the problem (i.e., the optimal objective value of the LP is an upper bound on OPT).

$$\begin{array}{lll} \text{maximize} & \sum_{ij\in E} b_{ij}x_{ij} & \\ \text{subject to} & \sum_{j\in M:ij\in E} b_{ij}x_{ij} \leq B_i & \forall i \in N, \\ & \sum_{i\in N:ij\in E} l_i x_{ij} \leq L_j & \forall j \in M, \\ & 0 \leq x_{ij} \leq 1 & \forall ij \in E. \end{array} \tag{3.1}$$

In this formulation, variable x_{ij} represents what fraction of b_{ij} contributes to the revenue. The first constraint represents that the total payment for an agent i does not exceed their budget B_i. The second constraint is the capacity constraint on an item.

For the MBA problem, this relaxation is equivalent to the one used for deriving approximation algorithms in [1, 4]. These previous studies prove the fact that the LP is a relaxation of the MBA problem. We can prove in the same way that (3.1) relaxes the knapsack MBA problem, and thus we omit the details.

Our algorithm computes an allocation by rounding an optimal solution x for (3.1). We say that an agent $i \in N$ (resp., an item $j \in M$) is *tight* with respect to a solution x if the

corresponding constraint in the relaxation holds with equality, i.e., $\sum_{j\in M:ij\in E} b_{ij}x_{ij} = B_i$ (resp., $\sum_{i\in N:ij\in E} l_i x_{ij} = L_j$).

Before proceeding, we first state a property of the LP relaxation. The proof of the following lemma is similar to that used in [4].

Lemma 3.1. *There exists an optimal solution x^* to (3.1) such that $E^* := \{ij \in E \mid 0 < x^*_{ij} < 1\}$ forms a forest on the vertex set $N \cup M$. In each connected component of the forest, there exists at most one non-tight vertex. Such an optimal solution x^* can be computed in polynomial time.*

Proof. Let x be an extreme point optimal solution for (3.1), and let $E_x = \{ij \in E \mid 0 < x_{ij} < 1\}$. Suppose that the vertices in $N' \subseteq N$ and in $M' \subseteq M$ form a connected component of the graph $(N \cup M, E_x)$. We need to show that this connected component is a tree that has at most one non-tight vertex.

Let $n' = |N'|$ and $m' = |M'|$, and let E' be the edges in E_x induced by $N' \cup M'$. Besides the constraints $0 \le x_{ij} \le 1$ $(ij \in E)$, (3.1) contains $n' + m'$ constraints corresponding to the agents in N' and the items in M'. From the basic properties of linear algebra, $|E'|$ is at most the number of tight constraints among these $n' + m'$ constraints; a similar claim can be found in [4], and more general argument is proven in [10, Lemma 2.1.4]. Hence we have $|E'| \le n' + m'$, implying that there is at most one cycle in the connected component. Simultaneously, $|E'| \ge n' + m' - 1$ holds because $(N' \cup M', E')$ is connected, implying that at most one vertex in $N' \cup M'$ is not tight. If there is a cycle, then all of the $n' + m'$ vertices are tight.

If the connected component does not have a cycle, then we are done. Suppose that there exists a cycle on vertices $i_1, \ldots, i_k \in N'$ and $j_1, \ldots, j_k \in M'$ and that the cycle consists of edges $j_1i_1, i_1j_2, \ldots, j_ki_k, i_kj_1 \in E'$. Then we decrease $x_{j_ti_t}$ and increase $x_{i_tj_{t+1}}$ by a certain amount for each $t \in \{1, \ldots, k\}$, where we let j_{k+1} denote j_1 for notational convenience. The increase and decrease of the variables are decided as follows. Let ε be a positive number. $x_{j_1i_1}$ is decreased by $\varepsilon_{j_1i_1} := \varepsilon$. More generally, let $\varepsilon_{j_ti_t}$ denote the decrease of $x_{j_ti_t}$ for some $t \in [k]$. Then $x_{i_tj_{t+1}}$ is increased by $\varepsilon_{i_tj_{t+1}} := b_{j_ti_t}\varepsilon_{j_ti_t}/b_{i_tj_{t+1}}$. If the increase of $x_{i_tj_{t+1}}$ is $\varepsilon_{i_tj_{t+1}}$, then $x_{j_{t+1}i_{t+1}}$ is decreased by $\varepsilon_{j_{t+1}i_{t+1}} := l_{i_t}\varepsilon_{i_tj_{t+1}}/l_{i_{t+1}}$. Without loss of generality, we assume that $l_{i_k}\varepsilon_{i_kj_1} \le l_{i_1}\varepsilon_{j_1i_1}$ (otherwise, we change the direction of the cycle). After this operation, all of the constraints are still satisfied. Moreover, they are tight except the one corresponding to j_1 (when $l_{i_k}\varepsilon_{i_kj_1} < l_{i_1}\varepsilon_{j_1i_1}$, the constraint corresponding to j_1 is not tight after the operation). Let x' denote the solution after the operation. We set ε to the largest value such that the constraint $0 \le x'_{ij} \le 1$ still holds for each $ij \in E'$. x' is another feasible solution giving the same objective value as the original solution x. In other words, x' is an optimal solution to (3.1). By the setting of ε, at least one edge e in $\{i_tj_t, j_ti_{t+1} \mid t \in [k]\}$ satisfies $x'_e = 0$ or $x'_e = 1$. Let $E'' = \{ij \in E' \mid 0 < x'_{ij} < 1\}$. Then, $(N' \cup M', E'')$ has no cycles, and so is a forest (it is not a tree when more than one edge e on the cycle satisfies $x'_e = 0$ or $x'_e = 1$). In the construction of x' from x, the tightness of all vertices except j_k is maintained. Hence, $N' \cup M'$ contains at most one non-tight vertex.

Note that the above modification of x to x' can be performed in polynomial time. We obtain a required optimal solution x^* to (3.1) by applying this modification to each cycle in graph $(N \cup M, E_x)$, which is possible in polynomial time. We also note that there is a polynomial-time algorithm for computing an extreme point optimal solution to an LP. Therefore x^* can be solved in polynomial time. □

Our algorithm is based on the iterative rounding method of the LP relaxation, which computes an approximate solution by iterating a certain rounding procedure. In each itera-

tion, our algorithm computes an optimal solution x^* to (3.1) given in Lemma 3.1. As stated in the lemma, each connected component of $G := (N \cup M, E^*)$ has at most one non-tight vertex. By a simple case analysis, we can observe that these connected components have the property stated in the following lemma.

Lemma 3.2. *Let C be a connected component of G. If C consists of more than one vertex, then it includes an agent $i \in N$ that satisfies one of the following conditions:*

Case 1: *agent i is a tight leaf;*
Case 2: *agent i is tight and all neighbors of agent i except one are leaves;*
Case 3: *C is a star with agent i as its center.*

Proof. Consider a longest path of C. If this path contains only one agent, then C is a star and this agent is the center of the star, which means that we are in Case 3. If the path contains more than one agent, then two of the agents are leaves or neighbors of leaves. Moreover, at least one of these two agents is tight. Thus we are in Case 1 or 2. □

Our algorithm rounds variables that appear in the constraint corresponding to such an agent i when E^* is non-empty. The rounding procedure is included in Algorithm 1.

Algorithm 1: Bicriteria $(1-r, 2)$-approximation algorithm

Input : N, M, $B_i \in \mathbb{R}_+$ $(i \in N)$, $b_{ij} \in \mathbb{R}_+$ $(i \in N, j \in M)$, $l_i \in \mathbb{R}_+$ $(i \in N)$, $L_j \in \mathbb{R}_+$ $(j \in M)$
Output: $S_j \subseteq N$ $(j \in M)$

$E \leftarrow \{ij : i \in N, j \in M, l_i \leq L_j\}$, $S_j \leftarrow \emptyset$ for $\forall j \in M$;
while $E \neq \emptyset$ **do**
  compute an optimal solution x^* to (3.1) as described in Lemma 3.1;
  for $\forall ij \in E$ **do**
    if $x^*_{ij} = 0$ **then** $E \leftarrow E \setminus \{ij\}$;
    if $x^*_{ij} = 1$ **then** $E \leftarrow E \setminus \{ij\}$, $S_j \leftarrow S_j \cup \{i\}$, $B_i \leftarrow B_i - b_{ij}$, $L_j \leftarrow L_j - l_i$;
  $E^* \leftarrow \{ij \in E \mid 0 < x^*_{ij} < 1\}$;
  if $E^* \neq \emptyset$ *and* $\exists i \in N$ *of Case 1* **then**
    $j \leftarrow$ the neighbor of i in $(N \cup M, E^*)$, $E \leftarrow E \setminus \{ij\}$
  else if $E^* \neq \emptyset$ *and* $\exists i \in N$ *of Case 2* **then**
    $S_j \leftarrow S_j \cup \{i\}$ for each leaf neighbor j of i;
    $E \leftarrow E \setminus \{ij' \in E \mid j' \in M\}$
  else if $E^* \neq \emptyset$ *and* $\exists i \in N$ *of Case 3* **then**
    $S_j \leftarrow S_j \cup \{i\}$ for $\forall ij \in E^*$;
    $E \leftarrow \emptyset$
output $\{S_j : j \in M\}$

Theorem 3.1. *Algorithm 1 is a $(1-r, 2)$-approximation algorithm that runs in polynomial time for the knapsack MBA problem.*

Proof. In each iteration, the algorithm removes at least one edge from E. Hence it runs in polynomial time. When the algorithm adds i to S_j, $x^*_{ij} = 1$ or item j is a leaf. In the former case, the algorithm simultaneously decreases the capacity of item j by l_i. Hence the total length of agents allocated item j in the former case is at most the original capacity of item

j. The latter case occurs at most once for each item j, and when it does, the total length is increased by $l_i \leq L_j$. Hence, for each $j \in M$, the total length of agents allocated item j is at most twice the original capacity of item j at the termination of the algorithm.

Turning to the approximability of the revenue, let LP denote the optimal objective value of (3.1) at the beginning of the algorithm, and ALG denote the objective value attained by the output solution. Let $x^{(k)}$ be the optimal solution to (3.1) computed in the k-th iteration, and let $E^{(k)}$ denote the edge set E at the beginning of the k-th iteration. We denote the restriction of $x^{(k)}$ onto $E^{(k+1)}$ by $\bar{x}^{(k)}$. Moreover, we let $v^{(k)}$ and $\bar{v}^{(k)}$ be the objective values of $x^{(k)}$ and $\bar{x}^{(k)}$, respectively, and let $\Delta_k = v^{(k)} - \bar{v}^{(k)}$ (≥ 0). $\bar{x}^{(k)}$ is feasible for (3.1) solved in the $(k+1)$-th iteration, and hence $v^{(k+1)} \geq \bar{v}^{(k)}$. Therefore, $\mathrm{LP} \leq \sum_{k=1}^{K}(v^{(k)} - v^{(k+1)}) \leq \sum_{k=1}^{K}(v^{(k)} - \bar{v}^{(k)}) = \sum_{k=1}^{K}\Delta_k$, where K is the total number of iterations and $v^{(K+1)}$ is defined as 0.

Each edge $ij \in E^{(k)} \setminus E^{(k+1)}$ is removed from E in the k-th iteration because $x^{(k)}(ij) \in \{0,1\}$ or agent i satisfies Case 1, 2, or 3. If $x^{(k)}(ij) \in \{0,1\}$ or agent i satisfies Case 3, then the revenue of the solution maintained in the algorithm is increased by at least Δ_k. If agent i satisfies Case 1 or 2, then only one edge, say ij', incident to agent i in $E^{(k)}$ is removed from E without allocating agent i item j'. If we allocate agent i item j', then the revenue of the solution is increased by at least Δ_k in the k-th iteration. Let F denote the set of all such edges ij' that appear over the course of the algorithm. Then, $\mathrm{ALG} + \sum_{ij' \in F} b_{ij'} \geq \sum_{k=1}^{K}\Delta_k$ holds.

Let $N' = \{i \in N \mid ij' \in F\}$. We will show that $\sum_{i \in N'} B_i \leq \sum_{k=1}^{K}\Delta_k$ holds, where B_i denotes the budget of agent i at the beginning of the algorithm. When the k-th iteration of the algorithm decreases the budget of agent $i \in N'$ in Line 6 because of the existence of $j \in M$ with $x^*_{ij} = 1$, this decrease is included in Δ_k. When the edge $ij' \in F$ is removed in the k'-th iteration, the agent i is tight and all edges incident to agent i in E are removed from E. In this case, the budget of agent i remaining at the beginning of the k'-th iteration is included in $\Delta_{k'}$. These facts imply $\sum_{i \in N'} B_i \leq \sum_{k=1}^{K}\Delta_k$.

Notice that no agent $i \in N'$ has more than one incident edge in F. This and $\sum_{i \in N'} B_i \leq \sum_{k=1}^{K}\Delta_k$ together imply

$$\sum_{ij' \in F} b_{ij'} \leq \sum_{i \in N'}\left(B_i \max_{j \in M}\frac{b_{ij}}{B_i}\right) \leq r \cdot \sum_{k=1}^{K}\Delta_k.$$

Therefore, we have

$$\mathrm{ALG} \geq \sum_{k=1}^{K}\Delta_k - \sum_{ij' \in F} b_{ij'} \geq (1-r)\cdot\sum_{k=1}^{K}\Delta_k \geq (1-r)\cdot\mathrm{LP}.$$

Since the maximum revenue attained by any feasible allocation is at most LP, this proves the $(1-r)$-approximation of the revenue. □

Algorithm 1 can be modified to a $(1-r)/2$-approximation algorithm as follows. Let S_j be the set of agents allocated item j and let i_j be the last agent allocated item j in Algorithm 1. If the capacity constraint of item j is violated by S_j, then $S_j \setminus \{i_j\}$ satisfies the capacity constraint of item j. Define two allocations $S'_j := S_j \setminus \{i_j\}$ and $S''_j := \{i_j\}$ of item j, where we let $S'_j = S''_j = \emptyset$ if $S_j = \emptyset$. Then both S'_j and S''_j are feasible for the capacity constraint of item j. One of the allocations $\{S'_j : j \in J\}$ and $\{S''_j : j \in J\}$ achieves revenue of at least $(1-r)/2$. Hence, the algorithm that outputs the better of these two allocations is a $(1-r)/2$-approximation algorithm.

Corollary 3.1. *There exists a polynomial* $(1-r)/2$*-approximation algorithm for the knapsack MBA problem. In particular, the revenue of the allocation output by this algorithm is at least* $(1-r)/2$ *times the optimal objective value of* (3.1).

Note that the approximation ratio $(1-r)/2$ is inferior to the approximation ratio $(1-r)(1-1/(1+r)^{1/r})$ given by Sumita et al. [15]. However, the corollary is worth mentioning because they use different LP relaxations.

Let us remark briefly on the relationship between our algorithms and the $(1-r/4)$-approximation algorithm given by Chakrabarty and Goel [4] for the MBA problem. A claim similar to Lemma 3.1 appears in the analysis in [4] but is slightly stronger: Chakrabarty and Goel showed that the edge set $\{ij \in E \mid 0 < x^*_{ij} \leq 1\}$ forms a forest, whereas Lemma 3.1 insists that the edge set $\{ij \in E \mid 0 < x^*_{ij} < 1\}$ does so. It is not difficult to see that their claim cannot be extended to the knapsack MBA problem. Because of this difference, we had to weaken the approximation guarantee.

4. Pseudo-Polynomial $1/3 \cdot (1-r/4)$-Approximation Algorithm

In this section, we propose a pseudo-polynomial approximation algorithm. Our algorithm relies on an idea used in the $(1-r/4)$-approximation algorithm for the MBA problem given by Kalaitzis et al. [9]. Their algorithm is motivated by Shmoys and Tardos' algorithm for the generalized assignment problem [13]. First, we introduce a key claim used in their algorithm.

4.1. Key property in the MBA problem

Let $i \in N$. Suppose that $x\colon N \times M \to [0,1]$ satisfies $\sum_{j\in M} b_{ij}x_{ij} \leq B_i$. From x, we generate a random subset T_i of M as follows. We assume without loss of generality that $b_{i1} \geq b_{i2} \geq \cdots \geq b_{im}$. Let $K = \lceil \sum_{j\in M} x_{ij} \rceil$ and define K buckets $\beta_1, \ldots, \beta_K$. We allocate each item j to buckets fractionally so that

- the total fraction of j allocated to buckets is x_{ij},
- each of buckets $\beta_1, \ldots, \beta_{K-1}$ is allocated exactly one unit of items (and hence the bucket β_K is allocated at most one unit of items),
- and an item of smaller index is allocated to buckets of smaller index preferentially.

For each $j \in M$ and $k \in [K]$, we let $y^{(i)}_{kj}$ denote the fraction of item j allocated to the k-th bucket. In other words, $y^{(i)}$ is the output of Algorithm 2.

Algorithm 2: Procedure to compute $y^{(i)}$

```
1 j ← 1, k ← 1, x'_{j'} ← x_{ij'} (j' ∈ M), c_{k'} ← 1 (k' = 1, ..., K), and y^{(i)}_{k'j'} ← 0 (k' ∈ [K], j' ∈ M);
2 while j ≤ m do
3 |   update y^{(i)}_{kj} ← min{x'_j, c_k}, x'_j ← x'_j − y^{(i)}_{kj}, and c_k ← c_k − y^{(i)}_{kj};
4 |   if c_k = 0 then update k := k + 1;
5 |   else j := j + 1;                                   // x'_j = 0
6 output y^{(i)} and terminate
```

By construction, $y^{(i)}$ satisfies the following conditions:

- $\sum_{j\in M} y^{(i)}_{kj} = 1$ holds for any $k \in [K-1]$, and $\sum_{j\in M} y^{(i)}_{Kj} \leq 1$;
- $\sum_{k=1}^{K} y^{(i)}_{kj} = x_{ij}$ holds for any $j \in M$.

Let $\alpha \geq 1$ be some constant. We consider the complete bipartite graph H with the bipartition $\{[K], M\}$ of the vertex set. Let F be a random matching in H such that $\Pr[kj \in F] = y_{kj}^{(i)}/\alpha$ for each $k \in [K]$ and $j \in M$. We define T_i as the set of items corresponding to the vertices matched by F. By construction, each item j is included in T_i with probability x_{ij}/α.

Kalaitzis et al. [9] proved the following fact (the original claim by Kalaitzis et al. considers only the case of $\alpha = 1$, but its proof can be modified for any $\alpha \geq 1$). For completeness, we provide its proof.

Lemma 4.1 (Kalaitzis et al. [9])**.** *Let F be a random matching in the bipartite graph H such that $\Pr[kj \in F] = y_{kj}^{(i)}/\alpha$ for each $k \in [K]$ and $j \in M$, and T_i be the set of items corresponding to the vertices matched by F. Then,*

$$\mathbb{E}\left[\min\left\{\sum_{j\in T_i} b_{ij}, B_i\right\}\right] \geq \frac{1}{\alpha}\left(1 - \frac{r}{4}\right)\sum_{j\in M} b_{ij}x_{ij}.$$

Proof. As noted above, $\sum_{j\in M} y_{kj}^{(i)} = 1$ holds for all buckets $k \in [K-1]$, but we have only $\sum_{j\in M} y_{Kj}^{(i)} \leq 1$ for the K-th bucket. In this proof, we assume $\sum_{j\in M} y_{Kj}^{(i)} = 1$ holds; otherwise, we make the equality hold by adding a dummy item j with $b_{ij} = 0$ and $x_{ij} = 1 - \sum_{j\in M} y_{Kj}^{(i)}$.

Since F is a matching, it matches each bucket with at most one item. We say that bucket β_k pays b_{ij} if F matches β_k with item j, whereas the payment of β_k is 0 when the bucket is matched with no item. Since each bucket β_k and each item j are matched with probability $y_{kj}^{(i)}/\alpha$, the total payment of buckets is $\mathbb{E}[\sum_{j\in T_i} b_{ij}] = \sum_{k=1}^{K}\sum_{j\in M} b_{ij}y_{kj}^{(i)}/\alpha = \sum_{j\in M} b_{ij}x_{ij}/\alpha$ in expectation. Since we assumed $\sum_{j\in M} b_{ij}x_{ij} \leq B_i$, this is at most B_i/α. We suppose that it is exactly B_i/α' for $\alpha' \geq \alpha$, i.e., $\alpha' = \mathbb{E}[\sum_{j\in T_i} b_{ij}]/B_i$.

For each $k \in [K]$, let a_k be the average payment of bucket β_k, i.e., $a_k = \sum_{j\in M} b_{ij}y_{kj}^{(i)}/\alpha$. Notice that $\sum_{k\in[K]} a_k = \sum_{j\in M} b_{ij}x_{ij}/\alpha = B_i/\alpha'$ by the above discussion. For a bucket β_k matched with item $j \in M$, we define its truncated payment b'_{kj} as $\min\{b_{ij}, \alpha' a_k\}$. Then, we have

$$\mathbb{E}\left[\min\left\{\sum_{j\in T_i} b_{ij}, B_i\right\}\right] \geq \mathbb{E}\left[\sum_{kj\in F} b'_{kj}\right].$$

We will prove that the right-hand side of this inequality is at least $\left(1 - \frac{r}{4}\right)\sum_{k=1}^{K} a_k$ $(= \sum_{j\in M} b_{ij}x_{ij}/\alpha)$.

For each $k \in [K]$, let h_k (resp., v_k) denote the average payment of β_k conditioned on that (i) the payment is at least (resp., smaller than) $\alpha' a_k$ and (ii) β_k is matched with some item. Notice that the probability for (ii) is exactly $\sum_{j\in M} y_{kj}^{(i)}/\alpha = 1/\alpha$ for each buckets β_k. The probability for (i) conditioned on (ii) is $z_k = \sum_{j\in M: b_{ij}\geq \alpha' a_k} y_{kj}^{(i)}$. Then, we can write h_k and v_k as $h_k = \sum_{j\in M: b_{ij}\geq \alpha' a_k} b_{ij}y_{kj}^{(i)}/z_k$ and $v_k = \sum_{j\in M: b_{ij}< \alpha' a_k} b_{ij}y_{kj}^{(i)}/(1 - z_k)$. We have $v_k \geq h_{k+1}$ for any $k \in [K-1]$ because $b_{ij} \geq b_{ij'}$ for any $j \leq j'$ and items of smaller index are allocated to buckets of smaller index preferentially when y is defined.

Since the payment for each item j counted in h_k is truncated to $\alpha' a_k$ in b'_{kj}, we have

$\sum_{k=1}^{K} a_k - \mathbb{E}\left[\sum_{kj\in F} b'_{kj}\right] = \sum_{k=1}^{K}(h_k - \alpha' a_k)z_k/\alpha$. Note that $a_k = \frac{(h_k - v_k)z_k + v_k}{\alpha}$ holds. Hence,

$$\begin{aligned}\frac{(h_k - \alpha' a_k)z_k}{\alpha} &= \left(h_k - \frac{\alpha'}{\alpha}v_k\right)\frac{z_k}{\alpha} - (h_k - v_k)\frac{\alpha' z_k^2}{\alpha^2} \\ &\le \frac{(h_k - v_k)(z_k - \frac{\alpha'}{\alpha}z_k^2)}{\alpha} \\ &\le \frac{h_k - v_k}{4\alpha'},\end{aligned}$$

where the first inequality follows from $\alpha'/\alpha \ge 1$ and the second one follows from the fact that $z_k - \frac{\alpha'}{\alpha}z_k^2$ attains maximum value $\alpha/(4\alpha')$ when $z_k = \alpha/(2\alpha')$. If $k < K$, then $v_k \ge h_{k+1}$, and hence $(h_k - v_k)/(4\alpha') \le (h_k - h_{k+1})/(4\alpha')$. Summing,

$$\sum_{k=1}^{K} a_k - \mathbb{E}\left[\sum_{kj\in F} b'_{kj}\right] \le \frac{1}{4\alpha'}\left(\sum_{k=1}^{K-1}(h_k - h_{k+1}) + h_K - v_K\right) \le \frac{h_1}{4\alpha'}.$$

This implies $\mathbb{E}\left[\sum_{kj\in F} b'_{kj}\right] \ge \sum_{k=1}^{K} a_k - \frac{h_1}{4\alpha'} = \left(1 - \frac{h_1}{4B_i}\right)\sum_{k=1}^{K} a_k$. The required inequality follows because $h_1/B_i \le r$. □

4.2. Algorithm for the knapsack MBA

We use the following LP relaxation of the problem in our proposed algorithm.

$$\begin{array}{lll} \text{maximize} & \sum_{i\in N}\sum_{j\in M}\sum_{t\in[L_j - l_i]^+} b_{ij}x_{ijt} & \\ \text{subject to} & \sum_{j\in M}\sum_{t\in[L_j]^+} b_{ij}x_{ijt} \le B_i & \forall i \in N, \\ & \sum_{t\in[L_j]^+} x_{ijt} \le 1 & \forall i \in N, j \in M, \\ & \sum_{i\in N}\sum_{t'=\max\{t-l_i+1,0\}}^{t} x_{ijt'} \le 1 & \forall j \in M, t \in [L_j]^+, \\ & 0 \le x_{ijt} \le 1 & \forall ij \in E, t \in [L_j]^+. \end{array} \tag{4.1}$$

This relaxation formulates the capacity constraints for items by introducing a time notation. We consider the process that allocates each item j to start at time 0 and to have a time limit of L_j. If agent i is allocated item j, then the process allocating item j is occupied by agent i for l_i units of time. The capacity constraint for item j is satisfied if the process finishes for all agents allocated item j before the item's time limit. $x_{ijt} = 1$ corresponds to the decision that agent i is allocated item j and the processing for this agent starts at time t. The third constraint of (4.1) requires that, at any time $t \in [L_j]^+$, the process is allocating item j for only one agent. Based on this interpretation, $\sum_{t\in[L_j]^+} x_{ijt}$ represents whether agent i is allocated item j. The first constraint represents the budget constraint for each agent i, and the second constraint requires that each agent be allocated each item at most once. Although x_{ijt} is defined even for $t > L_j - l_i$ for notational convenience, we can assume that it takes value 0 in an optimal solution because a positive value would not contribute to the objective function.

In our algorithm, we first compute an optimal solution x for (4.1). Then, we define a bipartite graph H' from x as follows. Let U be the set of pairs of $j \in M$ and $t \in [L_j]^+$. We regard x as the edge weights on the complete bipartite graph over the bipartition (N, U) of the vertex set. From x, we construct buckets $\beta_{i,1}, \ldots, \beta_{i,k_i}$ for each $i \in N$ as above (i.e., $k_i = \lceil \sum_{(j,t)\in U} x'_{ijt} \rceil$). Let P_i denote the set of buckets constructed from $i \in N$, and $P = \bigcup_{i\in N} P_i$. The bipartite graph H' is defined as the complete bipartite graph over the vertex sets P and U.

We denote an edge joining $p \in P$ and $(j,t) \in U$ by pjt. For $p \in P$, let l_p denote the length of the agent from which bucket p is constructed. We say that an edge pjt *intersects* another edge $p'jt'$ if $t \leq t' < t + l_p$ or $t' \leq t < t' + l_{p'}$.

Our algorithm chooses a matching of H', and constructs an allocation of items to agents from the matching. Here, we associate a matching with an allocation by making an edge pjt in the matching mean that item j is occupied by the agent of bucket p from time t exclusive to time $t + l_p$. Not all matchings of H' give allocations satisfying the capacity constraint, and thus we impose the following conditions on the matching:

(i) if $(j,t) \in U$ is matched with $p \in P_i$, then (j,t') is not matched with any buckets in P_i for all $t' \in [L_j]^+ \setminus \{t\}$;

(ii) if edge pjt is included in the matching, then no other edge $p'jt'$ intersecting pjt is included.

Condition (i) means that the matching includes at most one edge that implies allocating item j to agent i. Condition (ii) indicates that, if pjt is included in the matching, then no other agents occupy item j from time t to time $t + l_p$, where another agent is allowed to start its process at time $t + l_p$. We call a matching in H' *feasible* if it satisfies conditions (i) and (ii).

The characteristic vector of a matching F is $\chi \in \{0,1\}^{P \times U}$ such that $\chi(p,j,t) = 1$ if and only if edge pjt is included in F. We sometimes identify a matching and its characteristic vector.

Let $y^{(i)}$ be the fractional allocations of pairs $(j,t) \in U$ to the buckets in P_i introduced in Section 4.1, and let $y\colon P \times U \to [0,1]$ be the vector such that $y_{pjt} = y^{(i)}_{pjt}$ for all $i \in I$, $p \in P_i$, and $(j,t) \in U$. Vector y satisfies the following conditions:

(iii) for any $p \in P$, $\sum_{(j,t) \in U} y_{pjt} \leq 1$ holds (by the construction of the buckets);

(iv) for any $i \in N$ and $j \in M$, $\sum_{p \in P_i} \sum_{t \in [L_j]^+} y_{pjt} \leq 1$ holds (by the second constraint of (4.1));

(v) for any $j \in M$ and $t \in [T]^+$, $\sum_{p \in P} \sum_{t'=\max\{t-l_p+1,0\}}^{t} y_{pjt'} \leq 1$ (by the third constraint of (4.1)).

We will prove the following lemma.

Lemma 4.2. *Vector $y/3$ can be expressed as a convex combination of feasible matchings, and this convex combination can be computed in pseudo-polynomial time.*

Proof. When y is the zero-vector, then the lemma is trivial. We hence assume that y has at least one non-zero element, and construct a convex combination of feasible matchings inductively. Choose an arbitrary item $j \in M$, and define $p \in P$ and $t \in [L_j]^+$ so as to minimize $t + l_p$ subject to $y_{pjt} > 0$. We modify y by setting y_{pjt} to 0. In the following, the obtained vector is called y' while the original vector is called y. We suppose that $y'/3$ is represented as a convex combination $\sum_{F \in \mathcal{F}} w_F \chi_F$ of feasible matchings, where $\mathcal{F}$ is a set of feasible matchings, χ_F is the characteristic vector of $F \in \mathcal{F}$, and w_F is a weight of F (i.e., $w_F > 0$ for all $F \in \mathcal{F}$ and $\sum_{F \in \mathcal{F}} w_F = 1$). In the rest of this proof, we show how to construct a convex combination representing $y/3$ from $(\mathcal{F}, w)$.

Since $y'_{pjt} = 0$, no matching in $\mathcal{F}$ includes edge pjt. We call a matching $F \in \mathcal{F}$ *eligible* if pjt can be added to F without violating the feasibility. If the total weight of eligible matchings in $\mathcal{F}$ is at least $y_{pjt}/3$, then we obtain a convex combination for $y/3$ by adding pjt to these matchings; if the total weight of eligible matchings is larger than $y_{pjt}/3$, then we modify the matchings so that the total weight of matchings including pjt is exactly $y_{pjt}/3$. Notice that we can do this modification so that the number of matchings in the convex combination is increased by at most one.

Now we prove that the total weight of eligible matchings in $\mathcal{F}$ is at least $y_{pjt}/3$. Let i be the agent such that $p \in P_i$. A matching F is not eligible if one of the following conditions holds:

(vi) F includes an edge incident to p;

(vii) F includes an edge $p'jt'$ such that $p' \in P_i$ and $t' \in [L_j]^+$;

(viii) F includes an edge $p'jt'$ that intersects pjt.

The total weight of matchings satisfying condition (vi) is at most $\sum_{(j',t')\in U} y'_{pj't'}/3 = \sum_{(j',t')\in U} y_{pj't'}/3 - y_{pjt}/3 \leq 1/3 - y_{pjt}/3$, where the inequality follows from condition (iii). Similarly, the total weight of matchings satisfying condition (vii) is at most $1/3$ since we have $\sum_{p'\in P_i} \sum_{t'\in[L_j]^+} y_{p'jt'} \leq 1$ by condition (iv).

Let us bound the total weight of matchings satisfying condition (viii). Recall that we defined p and t so that $t + l_p$ is minimized subject to $y_{pjt} > 0$. Thus, if a matching $F \in \mathcal{F}$ includes $p'jt'$ (implying $y_{p'jt'} > 0$) and if $p'jt'$ intersects pjt, then $t' < t + l_p \leq t' + l_{p'}$ holds, which is equivalent to $t + l_p - l_{p'} \leq t' < t + l_p$. Condition (v) indicates that $\sum_{p'\in P} \sum_{t'=t+l_p-l_{p'}}^{t+l_p-1} y_{p'jt'} \leq 1$ holds (t in condition (v) is set to $t+l_p-1$ here). Thus the total weight of matchings satisfying condition (viii) is at most $1/3$.

Summing up, matchings which is not eligible have weight at most $1-y_{pjt}/3$ in total. Thus, the total weight of eligible matchings is at least $y_{pjt}/3$. As noted above, this implies that a convex combination of feasible matchings representing $y/3$ can be constructed from $(\mathcal{F}, w)$, and the number of feasible matchings in this combination is at most $|\mathcal{F}| + 1$. The running time for this construction is polynomial in $|\mathcal{F}|$ and the size of H' (i.e., $O(|P||U|)$), assuming that $(\mathcal{F}, w)$ is given. Recall that $(\mathcal{F}, w)$ is a convex combination representing $y'/3$, and the number of non-zero elements of y' is smaller than that of y. Thus, to construct a convex combination for $y/3$, it suffices to repeat the above construction $O(|P||U|)$ times. This gives a pseudo-polynomial time algorithm for computing a convex combination representing $y/3$. □

In our algorithm, we construct a convex combination of feasible matchings claimed in Lemma 4.2, and output the allocation corresponding to the best matching in the convex combination. This is summarized as Algorithm 3.

Algorithm 3: $1/3 \cdot (1 - r/4)$-Approximation Algorithm

1 compute an optimal solution x to (4.1);
2 for each $i \in N$, compute $y^{(i)}$ from x by Algorithm 2 and let y denote its concatenation;
3 represent $y/3$ as a convex combination of feasible matchings by Lemma 4.2;
4 output the allocation corresponding to the best feasible matching in the convex combination.

Theorem 4.1. *Algorithm 3 achieves* $1/3 \cdot (1 - r/4)$*-approximation.*

Proof. Let F be a random feasible matching sampled from the convex combination representing $y/3$. For each $p \in P$ and $(j, t) \in U$, edge pjt is included in F with probability $y_{pjt}/3$. Hence, by Lemma 4.1, the objective value achieved by F is at least $1/3 \cdot (1 - r/4) \cdot \sum_{i\in N} \sum_{j\in M} \sum_{t\in[L_j-l_i]^+} b_{ij}x_{ijt}$ in expectation. The output of the algorithm achieves an objective value not worse than that of F, and $\sum_{i\in N} \sum_{j\in M} \sum_{t\in[L_j-l_i]^+} b_{ij}x_{ijt}$ is an upper bound on the optimal objective value. Therefore, the theorem is proven. □

5. Conclusion

In this paper, we consider the knapsack MBA problem, which is obtained by introducing a capacity constraint on each item into the MBA problem. We propose two LP-rounding algorithms. Both of the LP relaxations used in the algorithms formulate the budget constraints of agents as single inequalities. For the MBA problem, such LP relaxation has an integrality gap of at most 3/4 for any r, and algorithms breaking this limit are obtained from the configuration LP [8, 9]. A similar configuration LP can be also defined for the knapsack MBA problem, and indeed the online algorithm of Sumita et al. [15] uses it. One natural direction of future studies is to investigate the integrality gap of this LP.

Acknowledgments

This study is supported by JSPS KAKENHI Grant Number JP17K00040 and JST PRESTO Grant Number JPMJPR1759. The author is grateful to the anonymous reviewers for comments on earlier version of this paper.

References

[1] N. Andelman, Y. Mansour: Auctions with budget constraints. *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory* (2004), 26–38.

[2] N. Buchbinder, K. Jain, J. Naor: Online primal-dual algorithms for maximizing ad-auctions revenue. *Proceedings of the 15th Annual European Symposium on Algorithms* (2007), 253–264.

[3] G. Călinescu, C. Chekuri, M. Pál, J. Vondrák: Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, **40** (2011), 1740–1766.

[4] D. Chakrabarty, G. Goel: On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap. *SIAM Journal on Computing*, **39** (2010), 2189–2211.

[5] C. Chekuri, S. Khanna: A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, **35** (2005), 713–728.

[6] C. Chekuri, J. Vondrák, R. Zenklusen: Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, **43** (2014), 1831–1879.

[7] M. Dawande, J. Kalagnanam, P. Keskinocak, F.S. Salman, R. Ravi: Approximation algorithms for the multiple knapsack problem with assignment restrictions. *Journal of Combinatorial Optimization*, **4** (2000), 171–186.

[8] C. Kalaitzis: An improved approximation guarantee for the maximum budgeted allocation problem. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms* (2016), 1048–1066.

[9] C. Kalaitzis, A. Madry, A. Newman, L. Polácek, O. Svensson: On the configuration LP for maximum budgeted allocation. *Mathematical Programming*, **154** (2015), 427–462.

[10] L.C. Lau, R. Ravi, M. Singh: Iterative Method in Combinatorial Optimization. Cambridge University Press, 2011.

[11] A. Mehta, A. Saberi, U.V. Vazirani, V.V. Vazirani: Adwords and generalized online matching. *Journal of the ACM*, **54** (2007), 22.

[12] A. Schrijver: Theory of Linear and Integer Programming. John Wiley & Sons, 1998.

[13] D.B. Shmoys, E. Tardos: An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, **62** (1993), 461–474.

[14] A. Srinivasan: Budgeted allocations in the full-information setting. *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings* (2008), 247–253.

[15] H. Sumita, Y. Kawase, S. Fujita, T. Fukunaga: Online optimization of video-ad allocation. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (2017), 423–429.

Takuro Fukunaga
Faculty of Science and Engineering
Chuo University
Kasuga 1-13-27, Bunkyo-ku, Tokyo, 112-8551, Japan
E-mail: fukunaga.07s@g.chuo-u.ac.jp