The Generalized Maximum Coverage Problem

Reuven Cohen Liran Katzir
Department of Computer Science
Technion – Israel Institute of Technology
Haifa 32000, Israel

Abstract

We define a new problem called the Generalized Maximum Coverage Problem (GMC). GMC is an extension of the Budgeted Maximum Coverage Problem, and it has important applications in wireless OFDMA scheduling. We use a variation of the greedy algorithm to produce a $(\frac{2e-1}{e-1}+\epsilon)$-approximation for every $\epsilon > 0$, and then use partial enumeration to reduce the approximation ratio to $\frac{e}{e-1}+\epsilon$.

Keywords: maximum coverage, approximation algorithms

1 Introduction

We study the following new maximization problem, referred to as *the Generalized Maximum Coverage Problem* (GMC):

Instance: A budget L, a set $\mathbb{E}$ of elements, and a collection $\mathbb{B}$ of bins. For each bin $b \in \mathbb{B}$, each element $e \in \mathbb{E}$ has a positive profit and a non-negative weight, denoted by $P(b,e)$ and $W(b,e)$, respectively. In addition, each bin b has a weight $W(b)$, which can be considered as the overhead for using this bin.

Objective: Find a feasible selection with maximum profit. A selection is a triple $S = (\beta, \eta, f)$ where $\beta \subseteq \mathbb{B}$, $\eta \subseteq \mathbb{E}$, and f is an assignment function from η to β. The selection's weight is $W(S) = \sum_{b\in\beta} W(b) + \sum_{e\in\eta} W(f(e),e)$. A feasible selection is a selection such that $W(S) \leq L$. The selection's profit is denoted by $P(S)$, where $P(S) = \sum_{e\in\eta} P(f(e),e)$. Note that function f guarantees that an element e can only be chosen for one bin.

Note that the bins have no individual capacity. Instead, there is an overall budget L that limits the usage of bins and elements within the packing.

The new problem is related to three known problems: the Maximum Coverage Problem (MC) [4], the Budgeted Maximum Coverage Problem (BMC) [6], and the Multiple Choice Knapsack Problem (MCKP) [5] (see Figure 1). In MC, the input is a number L, a set of elements $\mathbb{E}$, and a collection $\mathcal{C}$ of subsets of $\mathbb{E}$. The goal is to find a subset of $\mathcal{C}$ whose size is $\leq L$ and which maximizes the number of elements covered. MC can be captured by the GMC formulation in the following way:

Problem	Restrictions	Best Known Approx
MC	$\forall_{b\in\mathbb{B}} W(b) = 1$, $\forall_{e\in\mathbb{E},b\in\mathbb{B}} P(b,e) = 1$, Elements can be selected only for a subset of the bins	$\frac{e}{e-1}$[4]
BMC	$\forall_{e\in\mathbb{E},b\in\mathbb{B}} P(b,e) = 1$, Elements can be selected only for a subset of the bins	$\frac{e}{e-1}$[6]
MCKP	$\forall_{b\in\mathbb{B}} W(b) = 0$	FPTAS [5]
GMC	none	$\frac{e}{e-1} + \epsilon,\ \forall\epsilon > 0$ (this paper)

Figure 1: The variations of MC and their upper bounds and restrictions.

(1) the bins in GMC represent the collection of subsets; (2) the weights of all bins are exactly 1; (3) the profit of every element is the same for all bins, and it is equal to 1; and (4) if an element e is a member of a collection C, then in its corresponding bin b we have $W(b,e) = 0$ and $W(b,e) = \infty$ otherwise. By setting $W(b,e) = \infty$ we prevent the selection of element e for bin b, while by setting $W(b,e) = 0$ we allow the selection of e for bin b with no additional cost. BMC is a generalization of MC in which the weight of the bins is not necessarily equal. GMC, defined in this paper, is a generalization of BMC, in which elements have weights and an element's weight/profit may vary from bin to bin. MC is NP-hard [4], and therefore both BMC and GMC are also NP-hard. A simple greedy algorithm yields an $(\frac{e}{e-1})$-approximation for MC [4]. The special case where an element can be selected for no more than ρ bins can be approximated within $\frac{1}{1-\left(1-\frac{1}{\rho}\right)^{\rho}}$ [1]. For small values of ρ, this approximation ratio is better than $\frac{e}{e-1}$. Several special cases where the bins are vertices of a graph and the elements are its edges are discussed in [7]. In [6], the authors prove a lower bound of $\frac{e}{e-1} - \epsilon$, for every $\epsilon > 0$, on the approximation ratio of MC. This lower bound also applies to BMC and GMC since they hold MC as a special case. The authors of [6] also present two greedy-based algorithms for BMC. The first achieves a $(\frac{\sqrt{e}}{\sqrt{e}-1})$-approximation and the second a $(\frac{e}{e-1})$-approximation (the best possible). If the bins have no weight, i.e., $\forall_{b\in\mathbb{B}} W(b) = 0$, GMC reduces to MCKP [5], which is known to have an FPTAS [5]. Therefore, the hardness of GMC stems from the bins' weights.

Another closely related related problem is the generalized assignment problem [2, 3, 9]. In this problem, each bin has no weight, but rather its own budget. The first known approximation algorithm for GAP is an LP-based 2-approximation algorithm, presented implicitly in [9]. An LP-based $(\frac{e}{e-1} + \epsilon)$-approximation algorithm is presented in [3]. In [2], the authors present a combinatorial linear time $(2+\epsilon)$-approximation.

Both BMC and MC are well established and have many applications [4]. An important application of GMC is in the area of OFDMA wireless scheduling. Consider a wireless base station with a set of packets waiting for transmission to different users. Each packet can be transmitted using several different Phy-profiles, each of which includes a modulation technique, code rate, and a scheme of forward error correcting code (FEC). There is a profit/cost tradeoff associated with selecting a Phy-profile for each packet. Loosely speaking, the more robust Phy-profiles require more bandwidth than the less robust ones. This added cost can be justified when the quality of a wireless link is bad and the robustness of the Phy-profile significantly increases the likelihood of a correct transmission.

For each frame transmitted by the base station, the scheduler needs to decide which of the packets will be transmitted, and what Phy-profile to use for each packet. All the packets transmitted using the same Phy-profile are grouped together and are considered as a single burst. In addition to the bandwidth cost associated with the transmission of every packet, there is also a fixed cost for every burst, because the base station needs to inform all the receiving nodes about the exact location of each burst within the frame. Due to the importance of this control information, it is transmitted using the most expensive and robust Phy-profile. Hence, the cost associated with each Phy-profile cannot be ignored.

In terms of GMC formulation, the frame's size is the budget L, each packet is an element, and each Phy-profile is a bin. The cost associated with the transmission of each packet using each Phy-profile is equivalent to the weight of the corresponding element in the corresponding bin. The profit associated with each packet indicates the likelihood that this packet will be transmitted correctly, multiplied by a factor that indicates the packet's priority at the considered time. The latter factor depends not on the Phy-profile, but only on the content of the packet.

In this paper we show how to generalize the BMC algorithms provided in [6] for GMC. The main new idea behind the proposed algorithms is not to use the original profits and weights for the entire execution of the algorithm, but to modify these values according to the constructing solution. As a result of this change, the entire analysis of the approximation ratio has to be extended, and a different argument is used for the concluding theorems.

The rest of this paper is organized as follows. In Section 2 we present a $(\frac{2e-1}{e-1}+\epsilon)$-approximation for GMC for every $\epsilon > 0$. In Section 3 we analyze the running time for this algorithm. In Section 4 we present an $(\frac{e}{e-1}+\epsilon)$-approximation for GMC for every $\epsilon > 0$, and in Section 5 we conclude the paper.

2 An Efficient Greedy-Based Approximation for GMC

The greedy algorithm presented in [4] for MC has L iterations. During every iteration, this algorithm picks the most profitable subset of elements in $\mathcal{C}$ from the not-yet-selected elements. This algorithm guarantees an $(\frac{e}{e-1})$-approximation [4]. An equivalent approach for BMC is to pick the highest density bin with respect to the set of not-yet-selected elements, until no bin can be added without violating the budget constraint. However, as shown in [6], such an algorithm has no worst-case approximation guarantee. In [6], it is also proven that a variation of this algorithm, which returns as a solution either the most profitable bin or the greedy solution, yields a $(\frac{\sqrt{e}}{\sqrt{e}-1})$-approximation for BMC. In this paper we use a similar approach for GMC, which achieves a $(\frac{2e-1}{e-1}+\epsilon)$-approximation[1] guarantee.

The main idea behind the proposed algorithm is to define the residual profit and residual weight of elements already selected for other bins. The *residual profit* is equal to the original profit minus the profit already gained when the element was selected. The *residual weight* is defined similarly, but with respect to the weights. The rationale behind this definition is that by selecting an element e for bin b_k instead of b_j, we gain only the profit difference and pay only the weight difference. This is stated formally as follows:

[1] $\frac{e}{e-1} \simeq 1.58$, $\frac{2e-1}{e-1} = 1 + \frac{e}{e-1} \simeq 2.58$, and $\frac{\sqrt{e}}{\sqrt{e}-1} \simeq 2.54$.

Definition 1 (residual profit/weight) *Consider a selection $S = (\beta, \eta, f)$, a bin b, and an element e. We define the residual profit $\boldsymbol{P_S(b, e)}$ to be equal to $P(b, e)$ if e is not selected by S (that is, $e \notin \eta$), or to $P(b, e) - P(f(e), e)$ otherwise. We define the residual weight $\boldsymbol{W_S(b, e)}$ in a similar way, but with respect to the weight rather than the profit. In addition, $\boldsymbol{W_S(b)} \triangleq W(b)$ if bin b is not selected by S (that is, $b \notin \beta$), and $W_S(b) \triangleq 0$ otherwise. Finally, for a selection $S' = (\beta', \eta', f')$, we define $\boldsymbol{P_S(S')}$ as $\sum_{e \in \eta'} P_S(f'(e), e)$, and $\boldsymbol{W_S(S')}$ as $\sum_{b \in \beta'} W_S(b) + \sum_{e \in \eta'} W_S(f'(e), e)$.* ■

The algorithm uses the residual profit and weight functions to choose a subset of elements from the same bin with approximately the highest residual density, until no elements can be added without violating the budget L. Elements with negative profit are ignored, regardless of their weight (even if it is negative). Subsets of elements with non-positive weights and positive profit are automatically chosen. When the above greedy phase ends, its selection is compared to the most profitable selection of a single bin. From these two possible selections, the one with the highest profit is chosen. The algorithm guarantees a $(\frac{2e-1}{e-1} + \epsilon)$-approximation for every $\epsilon > 0$.

Our algorithm employs two other algorithms, called ALG1 and ALG2. ALG1 is used for selecting a subset of elements with maximum residual density. ALG2 is used for selecting a subset of elements with maximum profit from a single bin. As explained later, finding an optimal solution for both sub-problems is also NP-hard. Therefore, the solutions found by ALG1 and ALG2 are only α-approximations where α can be arbitrarily close to 1. The exact details of ALG1 and ALG2 are discussed in Section 3. The core of our algorithm is the *Greedy*() procedure, presented below. As shown below, during the i'th iteration, *Greedy*() keeps a tentative selection S_i. To simplify the description of the algorithm, we now define the action of adding elements to this tentative selection:

Definition 2 (addition of elements) *For a selection $S = (\beta, \eta, f)$, a subset of elements E, and a bin b, we define by $\boldsymbol{S \oplus (b, E)}$ the addition of (b, E) to S. In other words, $S \oplus (b, E)$ is a new selection $S' = (\beta', \eta', f')$, where $\beta' = \beta \cup \{b\}$, $\eta' = \eta \cup E$, and*

$$f'(e) = \begin{cases} b, & \text{if } e \in E \\ f(e), & \text{otherwise.} \end{cases}$$

For a sequence $N = ((b^1, E^1), (b^2, E^2), \ldots, (b^q, E^q))$, we define by $\boldsymbol{S \oplus N}$ the result of successively adding the items of this sequence to S, that is, $S \oplus (b^1, E^1) \oplus (b^2, E^2) \oplus \cdots \oplus (b^q, E^q)$. ■

Definition 3 (residual profit/weight of added elements) *For a selection S, a subset of elements E, and a bin b, $\boldsymbol{P_S(b, E)} \triangleq P_S(S \oplus (b, E))$ and $\boldsymbol{W_S(b, E)} \triangleq W_S(S \oplus (b, E))$. Similarly, $\boldsymbol{P_S(N)} \triangleq P_S(S \oplus N)$, and $\boldsymbol{W_S(N)} \triangleq W_S(S \oplus N)$. Finally, $\boldsymbol{D_S(b, E)}$ is the residual density of E for bin b, namely, $\frac{P_S(b,E)}{W_S(b,E)}$.* ■

To simplify the notations, we use W_i, P_i and D_i to indicate the weight, profit, and density functions of tentative selection S_i. Namely, $\boldsymbol{W_i} = W_{S_i}$, $\boldsymbol{P_i} = P_{S_i}$, and $\boldsymbol{D_i} = D_{S_i}$.

Greedy(S_0):

1. $i \leftarrow 0$.
2. While new elements with a positive residual profit can be added to S_i without violating the budget L do:

(a) Invoke ALG1 to select a subset of elements E_i and a bin b_i such that $W_i(b_i, E_i) \leq L - W(S_i)$ and that (b_i, E_i) is an α-approximation with respect to the maximum density.

(b) $N_i \leftarrow nil$; iteratively concatenate to N_i pairs (b, E) with positive residual profit and non-negative weight, namely, $W_{S_i \oplus (b_i, E_i) \oplus N_i}(b, E) \leq 0$ and $\forall_{e \in E} P_{S_i \oplus (b_i, E_i) \oplus N_i}(b, e) > 0$. In other words, N_i contains a maximal ordered sequence of pairs that can be successively added to $S_i \oplus (b^i, E_i)$ without imposing any additional weight.

(c) $SE_i \leftarrow (b^i, E_i) || N_i$.

(d) $S_{i+1} \leftarrow S_i \oplus SE_i$.

(e) $i \leftarrow i + 1$.

3. Return (S_i).

Step 2(a) is the greedy part of the procedure. Step 2(b) is still crucial since a pair (b_i, E_i) with a negative residual weight but a positive residual profit has a negative residual density. Therefore, such a pair has a lower residual density than any pair with a positive, but very small residual profit and a huge weight. By adding a maximal sequence of pairs with a negative residual weight and a positive residual profit, we guarantee that during the next iteration the elements with the highest residual density will indeed be selected.

Algorithm 1

1. $S_G \leftarrow Greedy(\emptyset, \emptyset, \emptyset)$.
2. *Invoke ALG2 to find an α-approximation the most profitable selection S_H such that S_H selects exactly one bin and $W(S_H) \leq L$.*
3. *If the profit of S_G is greater than the profit of S_H, return (S_G). Else, return (S_H).*

We now prove that Algorithm 1 is a $(\frac{2e-1}{e-1} + \epsilon)$-approximation for GMC. We start with a sequence of five lemmas, which provide the foundation for Lemmas 6 and 7. Lemmas 6 and 7 can be viewed as generalizations of Lemmas 1 and 2 in [6]. Lemma 1 indicates that the residual profit/weight of a set of elements is just the sum of the residual profit/weight of these elements.

Lemma 1 *For a solution $S = (\beta, \eta, f)$, a subset of elements E, and a bin b, let $S' = (\beta', \eta', f') = S \oplus (b, E)$. Then, $P_S(b, E) = \sum_{e \in E} P_S(b, e)$ and $W_S(b, E) = W_S(b) + \sum_{e \in E} W_S(b, e)$.*

Proof: By Definition 3, $P_S(b, E) = P_S(S \oplus (b, E))$. By Definitions 1 and 2, if $S' = (\beta', \eta', f') = S \oplus (b, E)$ then, $P_S(S \oplus (b, E)) = \sum_{e \in \eta \cup E} P_S(f'(e), e)$.

$$\begin{aligned}
&\sum_{e \in \eta \cup E} P_S(f'(e), e) = \sum_{e \in E} P_S(b, e) + \sum_{e \in \eta \setminus E} P_S(f(e), e) \\
&= \sum_{e \in E} P_S(b, e) + \sum_{e \in \eta \setminus E} (P(f(e), e) - P(f(e), e)) \\
&= \sum_{e \in E} P_S(b, e).
\end{aligned}$$

The proof for the weight function W is similar and therefore omitted. ■

Lemma 2 indicates that the addition of a set E of elements to bin b increases the profit/weight of the tentative solution by the residual profit/weight of (b, E). Lemma 3 extends Lemma 2 for a sequence of additions. The proofs are straightforward, and are therefore omitted.

Lemma 2 *Let $S = (\beta_s, \eta_s, f_s)$, and let $S' = (\beta'_s, \eta'_s, f'_s) = S \oplus (b, E)$, where E is a subset of elements and b is a bin. Then, $P_S(b, E) = P(S') - P(S)$ and $W_S(b, E) = W(S') - W(S)$.*

Lemma 3 *Let $S = (\beta_s, \eta_s, f_s)$ a selection, $N = ((b^1, E^1), (b^2, E^2), \ldots, (b^n, E^n))$ and $S' = S \oplus N$. Then, $P_S(N) = P(S') - P(S)$ and $W_S(N) = W(S') - W(S)$.*

Corollary 1 *$P_i(SE_i) = P(S_{i+1}) - P(S_i)$ and $W_i(SE_i) = W(S_{i+1}) - W(S_i)$.*

Lemma 4 indicates that the total residual weight of the elements selected in step 2(a) has a positive increment.

Lemma 4 *For every $i \geq 0$, $W_i(b_i, E_i) > 0$ holds.*

Proof: The claim is proven by induction on the value of i. At the basis of the induction, we know that $W = W_0$. Since there are no elements with non-positive weights, $W_0(b_0, E_0) > 0$. For the inductive step $i > 0$, we have that $W_i(b_i, E_i) > 0$ by the induction hypothesis. After E_i is chosen, step 2(b) of the greedy procedure guarantees that if there exists a bin and elements pair (b, E) with $P_{S_i \oplus (b_i, E_i) \oplus N_i}(b, E) > 0$ and $W_{S_i \oplus (b_i, E_i) \oplus N_i}(b, E) \leq 0$, it is added to N_i. Therefore, $W_{i+1}(b_{i+1}, E_{i+1}) > 0$ holds as well. ■

Lemma 5 indicates that for two solutions S and S', the residual profit of S' with respect to S is bigger than the difference of their profits. This property will be used to compare the tentative solution to the optimal one, and to prove that there exists a sufficiently good augmentation to the tentative solution.

Lemma 5 *For every two selections $S = (\beta_s, \eta_s, f_s)$ and $S' = (\beta'_s, \eta'_s, f'_s)$, $P_S(S') \geq P(S') - P(S)$.*

Proof: Let $S^* = (\beta_s, \eta_s \cap \eta'_s, f^*_s)$, where $\forall_{e \in \eta_s \cap \eta'_s} f^*_s(e) = f_s(e)$. By Definition 1, we have $P_S(S') = \sum_{e \in \eta'_s} P_S(f'(e), e)$. Since only elements from η'_s contribute to this sum, $P_S(S') = P_{S^*}(S')$. Now,

$$\begin{aligned} P(S) &= \sum_{e \in \eta_s} P(f_s(e), e) \\ &= \sum_{e \in \eta_s \cap \eta'_s} P(f_s(e), e) + \sum_{e \in \eta_s \setminus \eta'_s} P(f_s(e), e) \\ &= P(S^*) + \sum_{e \in \eta_s \setminus \eta'_s} P(f_s(e), e). \end{aligned}$$

The last equation implies that $P(S) \geq P(S^*)$. Clearly, there is a sequence N such that $S' = S^* \oplus N$. By Lemma 3, $P_{S^*}(S') = P_{S^*}(S^* \oplus N) = P_{S^*}(N) = P(S') - P(S^*)$. Therefore, $P_{S^*}(S') \geq P(S') - P(S)$. ■

Definition 4 identifies the iteration for which a sufficiently good augmentation to the tentative solution might not exists. However, as seen later, this implies that for this iteration the tentative solution or a single bin solution is sufficiently good.

Definition 4 *Consider a feasible selection $S = (\beta, \eta, f)$. We define by $\boldsymbol{q(S)}$ the first iteration of Greedy() after which there exists in S at least one bin b such that adding (b, E) to $S_{q(S)}$, where $E = \{e \in \eta | f(e) = b\}$, would violate the budget constraint L. Namely, $W_{q(S)}(b, E) > L - W(S_{q(S)})$.* ■

The next two lemmas are the building block for proving the behavior of *Greedy*() during the first $q(S)$ iterations. These lemma enables us to determine the profit gained in comparison to a target solution S. Without loss of generality, we assume that S is not entirely contained in the greedy solution, since otherwise the greedy is at least as profitable as S. Recall that α is the approximation ratio for ALG1.

Lemma 6 *Let S be a selection. Then, the following holds for every i, where $0 < i < q(S)$:*

$$\frac{P(S_{i+1}) - P(S_i)}{W_i(b_i, E_i)} = \frac{P_i(SE_i)}{W_i(b_i, E_i)} \geq \frac{P_i(b_i, E_i)}{W_i(b_i, E_i)} \geq \frac{P_i(S)}{\alpha \cdot W_i(S)} \geq \frac{P(S) - P(S_i)}{\alpha \cdot W_i(S)}.$$

Proof: The first equality follows from Corollary 1. The first inequality follows from the fact that $P_i(SE_i)$ contains all items in (b_i, E_i) along with all items in N_i, and that each item in N_i has a positive residual profit. To prove the second inequality, let the average residual density of S with respect to S_i be $d = \frac{P_i(S)}{W_i(S)}$. Therefore, at least one of the bins in S, say b, has a density $\geq d$. The pair (b, E) is valid for addition to S_i since $q(S) > i$. ALG1 ensures that (b_i, E_i) is an α-approximation with respect to the maximal residual density. Therefore, we get $D_i(b_i, E_i) \geq \frac{d}{\alpha}$, which proves the third inequality. Finally, the last inequality $P_i(S) \geq P(S) - P(S_i)$ follows directly from Lemma 5. ■

Lemma 7 *Let S be a selection. Then, for every $0 < i < q(S)$*

$$P(S_{i+1}) \geq P(S) \cdot \left[1 - \prod_{k=0}^{i} \left(1 - \frac{W_k(b_k, E_k)}{\alpha \cdot W(S)}\right)\right].$$

Proof: The proof is by induction on the number of iterations. The induction basis is for $i = 0$. From Lemma 6 we have

$$\frac{P(S_1) - P(S_0)}{W_0(b_0, E_0)} \geq \frac{P_0(S)}{\alpha \cdot W_0(S)}.$$

Since $P \equiv P_0$, $W_0 \equiv W$ and $S_0 = (\emptyset, \emptyset, \emptyset)$, then by Lemma 4 we get

$$P(S_1) \geq P(S) \cdot \frac{W_0(b_0, E_0)}{\alpha \cdot W(S)}.$$

For the inductive step we assume that the claim holds for $j \leq i < q(S)$, and we prove it for $j = i + 1 < q(S)$. From Lemma 6 we get

$$\frac{P(S_{i+1}) - P(S_i)}{W_i(b_i, E_i)} \geq \frac{P(S) - P(S_i)}{\alpha \cdot W_i(S)}.$$

From Lemma 4 we get $W_i(b_i, E_i) > 0$. Thus,

$$P(S_{i+1}) \geq P(S_i) + W_i(b_i, E_i) \cdot \frac{P(S) - P(S_i)}{\alpha \cdot W_i(S)}.$$

We now develop the right side of this equation, while noting that $W_i(S) \leq W(S)$:

$$\begin{aligned}
&P(S_{i+1}) \geq P(S_i) + W_i(b_i, E_i) \cdot \frac{P(S) - P(S_i)}{\alpha \cdot W_i(S)} \\
&= P(S_i) \cdot \left(1 - \frac{W_i(b_i, E_i)}{\alpha \cdot W_i(S)}\right) + W_i(b_i, E_i) \cdot \frac{P(S)}{\alpha \cdot W_i(S)} \\
&\geq P(S) \cdot \left[1 - \prod_{k=0}^{i-1} \left(1 - \frac{W_k(b_k, E_k)}{\alpha \cdot W(S)}\right)\right] \\
&\cdot \left(1 - \frac{W_i(b_i, E_i)}{\alpha \cdot W_i(S)}\right) + W_i(b_i, E_i) \cdot \frac{P(S)}{\alpha \cdot W_i(S)} \\
&\geq P(S) \cdot \left[\left(1 - \frac{W_i(b_i, E_i)}{\alpha \cdot W_i(S)}\right) - \right. \\
&\left. \prod_{k=0}^{i} \left(1 - \frac{W_k(b_k, E_k)}{\alpha \cdot W(S)}\right)\right] + W_i(b_i, E_i) \cdot \frac{P(S)}{\alpha \cdot W_i(S)} \\
&= P(S) \cdot \left[1 - \prod_{k=0}^{i} \left(1 - \frac{W_k(b_k, E_k)}{\alpha \cdot W(S)}\right)\right].
\end{aligned}$$

The second inequality is the induction hypostasis on $P(S_i)$. ■

Lemma 8 uses lemma 7 to provide a worst-case analysis for the first $q(S)$ iterations.

Lemma 8 *Let $S = (\beta, \eta, f)$ be any selection. If $\sum_{k=0}^{i} W_k(b_k, E_k) \geq W(S)$ then $P(S_{i+1}) \geq P(S)\left(1 - e^{-\frac{1}{\alpha}}\right)$, where α is the approximation ratio of ALG1.*

Proof: Let b and A be two positive numbers and $a = a_0, a_1, \ldots, a_n$ be a sequence of positive numbers such that $\sum_{k=0}^{n} a_k \geq b \cdot A$. Then, it can be seen that the function $f(a) = \left[1 - \prod_{k=0}^{n}\left(1 - \frac{a_k}{A}\right)\right]$ receives its minimum at $a_k = \frac{b \cdot A}{n+1}$, and that $f(a) \geq \left[1 - \left(1 - \frac{b}{n+1}\right)^{n+1}\right] > 1 - e^{-b}$. The lemma then follows from combining this observation with Lemma 7, taking $A = \alpha \cdot W(S)$, $b = \frac{1}{\alpha}$, and $a_i = W_i(b_i, E_i)$. ■

Theorem 1 *Algorithm 1 is a $\left(\frac{1+\alpha-\alpha \cdot e^{-\frac{1}{\alpha}}}{1-e^{-\frac{1}{\alpha}}}\right)$-approximation for GMC.*

Proof: Let $S = (\beta, \eta, f)$ be the optimal selection. Let (b, E) be one of the pairs that define $q(S)$ (see Definition 4). Thus, $W(S_{q(S)}) + W_{q(S)}(b, E) > L$, and we have

$$\sum_{i=0}^{q(S)-1} W_i(b_i, E_i) \geq \sum_{i=0}^{q(S)-1} W_i(SE_i) = \sum_{i=0}^{q(S)-1} (W(S_{i+1}) - W(S_i)) = W(S_{q(S)}) > L - W_{q(S)}(b, E).$$

The first inequality stems from two observations: (1) by Lemma 3, for every i, $W_i(SE_i) = W_i(b_i, E_i) + W_{S_i \oplus (b_i,E_i)}(N_i)$ and (2) from the code of $Greedy()$, for every i, $W_{S_i \oplus (b_i,E_i)}(N_i) < 0$. The first equality stems from Corollary 1. The second equality is simple arithmetic, and the last inequality is due to the definition of $q(S)$.

Let $S' = (\beta', \eta', f')$ be a selection such that $\beta' = \beta \setminus b$, $\eta' = \eta \setminus E$, and $\forall_{e \in \eta'} f'(e) = f(e)$.

$$\begin{aligned} W(S_{q(S)}) &> L - W_{q(S)}(b, E) \\ &\geq W(S) - W_{q(S)}(b, E) \\ &= W(S') + W(b, E) - W_{q(S)}(b, E) \\ &\geq W(S'). \end{aligned}$$

The first inequality is from the previous equation. The second inequality holds because $L \geq W(S)$. The first equality holds because $W(S) = W(S') + W(b, E)$. Finally, the last inequality stems from the fact that $W(b, E) \geq W_{q(S)}(b, E)$. This is because the residual weight is always smaller than the original weight.

We conclude from the last two equations that $\sum_{i=0}^{q(S)-1} W_i(b_i, E_i) > W(S')$. Now, let $r = \frac{P(b,E)}{P(S)}$. Applying Lemma 8 using the previous inequality, we get

$$P(S_{q(S)}) > P(S') \cdot \left(1 - e^{-\frac{1}{\alpha}}\right) \geq P(S) \cdot (1 - r) \cdot \left(1 - e^{-\frac{1}{\alpha}}\right). \tag{1}$$

A selection that contains only elements from a single bin b is examined in step 2. The selection $S_{q(S)}$ is examined in step 1. Therefore, the returned selection is at least $P(S) \cdot \max\left(\frac{r}{\alpha}, (1 - r)\left(1 - e^{-\frac{1}{\alpha}}\right)\right)$. The minimum of this expression is $P(S)\frac{1-e^{-\frac{1}{\alpha}}}{1+\alpha-\alpha \cdot e^{-\frac{1}{\alpha}}}$, which means that the approximation ratio is $\frac{1+\alpha-\alpha \cdot e^{-\frac{1}{\alpha}}}{1-e^{-\frac{1}{\alpha}}}$. ■

An immediate implication is that by choosing a small enough α, the approximation ratio is $(\frac{2e-1}{e-1} + \epsilon)$ for every $\epsilon > 0$.

3 Run-Time Complexity Analysis

We now analyze the running time of Algorithm 1. The size of the input is $O(mn)$, where m is the number of elements and n is the number of bins. ALG2 simply solves (approximately) a Knapsack problem, since a most profitable subset of elements should be chosen under some budget constraint. To analyze the running time of $Greedy()$, note that in each iteration it chooses at least one bin-elements pair (in step 2(a)). Such a pair can also be selected in step 2(b). After a pair is added to the tentative selection, the residual profit of all elements in this pair for the considered bin is 0. Since no element with a non-positive profit can be chosen, the residual profit can only decrease. This implies that each bin-elements pair can only be selected once in steps 2(a) and 2(b). Thus each element can be selected at most m times during the entire run of the algorithm. Therefore, $Greedy()$ has $O(mn)$ iterations. Moreover, the amortized number of pairs in step 2(b) in all iterations is $O(mn)$. It is easy to see that all the operations in each iteration, except those in step 2, can also be performed in $O(mn)$.

To implement step 2(b) of $Greedy()$, we distinguish between two cases: (a) bin b is selected and an element has a positive profit and non-positive weight; (b) bin b is not selected, and there is a set of elements of positive profit and non-positive weight, and the total weight of b and E is non-positive. Both cases are easy to compute. In the first case we examine each element individually, and in the second we sum up all elements from bin b with positive profit and non-positive weight and compare it with b's weight.

In step 2(c) we need to add N_i to the tentative solution. Since the length of N_i is at most nm, this can be done by applying each bin-elements pair separately. In fact, we can apply such a pair immediately when it is identified, instead of accumulating the list of pairs. In such a case step 2(c) of the algorithm can be eliminated.

To address step 2(a) of $Greedy()$, we now show how to implement ALG1 such that the selected pair (b_i, E_i) will be an α-approximation for the maximum residual density pair (b_{max}, E_{max}) that satisfies $(b_{max}, E_{max}) < L - W(S_i)$. Formally, the residual density of a pair (b, E) during the i'th iteration of $Greedy()$ is

$$D_i(b, E) = \frac{P_i(b, E)}{W_i(b, E)} = \frac{\sum_{e \in E} P_i(b, e)}{W_i(b) + \sum_{e \in E} W_i(b, e)}.$$

If b was already selected in one of the previous iterations, then $W_i(b) = 0$. We know that $\forall_{e \in \mathbb{E}} W_i(b, e) > 0$ holds since otherwise e must have been selected to N_{i-1}. Therefore, the maximum of this weighted average is a single element whose residual density is maximum. If b was not selected ($W_i(b) > 0$) we know that $W_i(b, E) > 0$ holds for every subset of elements E with positive profits, since otherwise (b, E) must have been selected to N_{i-1}. In addition, if $W_i(b, e) \leq 0$ and $P_i(b, e) > 0$, then adding e will only increase the weighted density since it incurs no penalty and it increases the total profit. This means that an element e with $W_i(b, e) \leq 0$ and $P_i(b, e) > 0$ is present in every maximum density solution, and can be added to every pair automatically.

We now discuss the complexity of finding a maximum residual density set of elements for bin b in the case where $W_i(b) > 0$ and the residual weight of all other elements for this bin is positive. To this end, we define the following Maximum Density Knapsack (MDK) Problem:

Instance: A positive number c_0. A set S of items $s_1, s_2, \ldots, s_m$ and a capacity c. Each item s_i has positive profit $p(s_i)$ and a positive weight $w(s_i)$.

Objective: Find a subset $S' \subseteq S$ of items such that $\sum_{s_j \in S'} w(s_j) \leq c$ and the average density $\frac{\sum_{s_j \in S'} p(s_j)}{c_0 + \sum_{s_j \in S'} w(s_j)}$ is maximized.

MDK is NP-hard. This can be shown by a simple reduction from the Subset Sum Problem (SSP), which is known to be NP-hard [5]. We now show that there is an FPTAS for MDK. To this end, we consider the well-known Knapsack problem, defined as follows:

Instance: A capacity c. A set S of items $s_1, s_2, \ldots, s_m$ with profit $p(s_i)$ and a weight $w(s_i)$.

Objective: Find a subset $S' \subseteq S$ of items such that $\sum_{s_j \in S'} w(s_j) \leq c$ and $\sum_{s_j \in S'} p(s_j)$ is maximized.

To produce an FPTAS for MDK, we will use a scaling-based dynamic programming algorithm for Knapsack [5]. Denote by ϵ' the input for the FPTAS for Knapsack. This FPTAS ensures that for

every profit P, a minimum weight subset S' is found such that $P(S') \geq \left\lfloor \frac{P}{1+\epsilon'} \right\rfloor$. In the same way, denote by ϵ the input for the FPTAS for MDK. Let S_{opt} be a feasible set with maximum density. Since the Knapsack FPTAS finds a minimum weight set S' such that $P(S') \geq \left\lfloor \frac{P(S_{opt})}{1+\epsilon'} \right\rfloor$, we get that $W(S') \leq W(S_{opt}) \leq L$. Therefore,

$$\frac{P(S')}{c_0 + W(S')} \geq \frac{P(S')}{c_0 + W(S_{opt})} \geq \left\lfloor \frac{P(S_{opt})}{1+\epsilon'} \right\rfloor \cdot \frac{1}{c_0 + W(S_{opt})}.$$

We now choose ϵ' to satisfy

$$\left\lfloor \frac{P(S_{opt})}{1+\epsilon'} \right\rfloor \cdot \frac{1}{c_0 + W(S_{opt})} \geq \frac{1}{1+\epsilon} \cdot \frac{P(S_{opt})}{c_0 + W(S_{opt})}.$$

We now find the condition on ϵ' from this requirement:

$$\begin{aligned}
\left\lfloor \frac{P(S_{opt})}{1+\epsilon'} \right\rfloor &> \frac{P(S_{opt})}{1+\epsilon'} - 1 \\
\frac{P(S_{opt})}{1+\epsilon'} - 1 &\geq \frac{P(S_{opt})}{1+\epsilon} \\
\frac{P(S_{opt})}{1+\epsilon'} &\geq \frac{P(S_{opt}) + 1 + \epsilon}{1+\epsilon} \\
\frac{P(S_{opt})(1+\epsilon)}{P(S_{opt}) + 1 + \epsilon} &\geq 1 + \epsilon' \\
\frac{P(S_{opt}) + P(S_{opt})\epsilon}{P(S_{opt}) + 1 + \epsilon} &\geq 1 + \epsilon' \\
\frac{P(S_{opt})\epsilon - 1 - \epsilon}{P(S_{opt}) + 1 + \epsilon} &\geq \epsilon'.
\end{aligned}$$

If $\frac{1}{\epsilon} \geq \frac{P(S_{opt})}{c}$ for a constant c then $O(f(n,\epsilon))$ is polynomial in n and $P(S_{opt})$. So rather than using an approximate solution, we can use the optimal solution, which requires a polynomial in n and $P(S_{opt})$ running time, and have no error at all. Otherwise, when $P(S_{opt}) \gg \frac{1}{\epsilon}$, we get $\frac{P(S_{opt})\epsilon - 1 - \epsilon}{P(S_{opt}) + 1 + \epsilon} \approx \epsilon \geq \epsilon'$. Therefore, S' is a $(1+\epsilon)$-approximation for MDK. We can therefore conclude that MDK has an FPTAS.

Let $f(n,\epsilon)$ be the running time of a single FPTAS for Knapsack. The total running time for step 2(a) is $O(n \cdot f(n,\epsilon))$, and the total running time is $O(n^2 m \cdot f(n,\epsilon))$.

4 Improving the Performance Guarantee

In this section we show how to improve the performance guarantee from $\frac{2e-1}{e-1} + \epsilon$ to $\frac{e}{e-1} + \epsilon$ for every $\epsilon > 0$. The main idea is to apply the partial enumeration technique [8] before calling $Greedy()$. In this technique, a small but significant part of the optimal solution is "guessed", by iterating over the spectrum of all small solutions. For BMC [6], the authors used partial enumeration on the bins, which reduces the approximation ratio to $\frac{e}{e-1}$. However, even if all the bins of the optimal solution

are selected with this technique, the solution of *Greedy*() still has no performance guarantee. For example, consider an instance with one bin b and two elements $\{e_1, e_2\}$. Let $W(b) = 0$, $P(e_1) = 2$, $P(e_2) = L$, $W(e_1) = 1$ and $W(e_2) = L$. Not surprisingly, this example also shows that a greedy selection of the items for the Knapsack problem has no performance guarantee. This problem can be resolved by using another enumeration to guess elements as well.

Algorithm 2, presented below, employs enumeration to guess both the most profitable bins and the most profitable elements (with their assignment to bins of the optimal solution). The intuition is that if adding a pair (b, E) to a selection would violate the budget constraint L, then the profit of this pair is small. Specifically, for an integer K, all feasible initial selections which use K elements and K bins are examined. For the most profitable K elements and K bins, $P(b, E)$ is not more than $\frac{1}{K}$ of the optimum. Algorithm 2 replaces Algorithm 1 presented in Section 2. Note that *Greedy*() remains unchanged.

Algorithm 2 employs another algorithm, called ALG3. For a set of bins B, ALG3 finds a $(1+\epsilon')$-approximate maximum profit selection S that uses all the bins in B and only bins in B, such that $W(S) \leq L$. The details of ALG3 are discussed at the end of this section.

Algorithm 2

For a set of bins B, denote $W(B) = \sum_{b \in B}(W(b))$.

1. *For each set of bins B such that $|B| \leq K$ and $W(B) \leq L$, invoke ALG3 to find a $(1+\epsilon')$-approximation for maximum profit selection while using all the bins in B and only bins in B. From all these selections, let the maximal one be $H_{\leq K}$.*

2. *For each selection $S_0 = (\beta_0, \eta_0, f_0)$ such that $|\eta_0| = K$, $W(S_0) \leq L$, and $|\beta_0| \leq K + |\{f_0(e) \mid e \in \eta_0\}|$, run Greedy($S_0$). From all these selections, let the maximal one be $H_{\geq K}$.*

3. *If $P(H_{\leq K}) > P(H_{\geq K})$, return $H_{\leq K}$. Else, return $H_{\geq K}$.*

To prove the final theorem, we need to tighten the bound on $q(S)$ for which Lemmas 6 and 7 hold.

Definition 5 *Consider a feasible selection $S = (\beta, \eta, f)$, and let $S_0 = (\beta_0, \eta_0, f_0)$ be its initial selection. We define by $\boldsymbol{q'(S)}$ the first iteration of Greedy() during which one of the following conditions is fulfilled: (1) there exists in S a bin $b \in \beta \setminus \beta_0$ such that adding (b, E) to $S_{q'(S)}$, where $E = \{e \in \eta | f(e) = b\}$, would violate the budget constraint L; (2) there exists in S an element $e \in \eta$ such that adding $(f(e), \{e\})$ to $S_{q'(S)}$ would violate the budget constraint L.* ■

This definition extends Definition 4 in instances where S_0 is not empty. It turns out that Lemmas 6 – 8 apply also for q'. Lemma 6 applies since a maximum density pair for the bins selected is S_0 is a single element. The correctness of Lemmas 7 and 8 for $q'(S)$ follows from Lemma 6.

Theorem 2 *Algorithm 2 is an approximation algorithm for GMC whose performance guarantee is $\left(1 - e^{-\frac{1}{\alpha}}\right)^{-1} \left(1 - \frac{1}{K}\right)^{-1}$.*

Proof: Let $S = (\beta, \eta, f)$ be the optimal selection. If $|\beta| \leq K$, namely the number of bins in S is less than K, then β will be considered in step 1 of Algorithm 2 and a $(1+\epsilon')$-approximation of S will be found. Thus, $\epsilon' = \frac{1}{e-1}$ would yield an $(\frac{e}{e-1})$-approximation.

If $|\beta| > K$, consider the iteration of step 2 during which the K most profitable items and the K most profitable bins are selected. Since the K most profitable items might belong to the K most profitable bins, β_0 might take any value in $[K, 2K]$. If $|\beta_0|$ is set to be equal to $K+|\{f_0(e) \mid e \in \eta_0\}|$, then solutions with exactly $K+1$ to $2K-1$ bins are excluded. Let (b, E) be one of the pairs that defines $q'(S)$ (see Definition 5). Thus, $W(S_{q'(S)}) + W_{q'(S)}(b, E) > L$. Now, let $r = \frac{P(b,E)}{P(S)}$. Using arguments similar to those in Eq.1, we get Eq.2.

$$P(S_{q'(S)}) \geq P(S) \cdot (1-r) \cdot \left(1 - e^{-\frac{1}{\alpha}}\right). \tag{2}$$

We now argue that $P_{q'(S)}(b, E) \leq P(b, E) \leq \frac{P(S)}{K}$. To prove this, consider first the case where E contains several elements. In this case, by the definition of $q'(S)$, bin b cannot be one of the K bins in S_0. Then, since S_0 holds the K most profitable bins, we get that any bin is less profitable than their average, which is bounded by $\frac{P(S)}{K}$. Next, consider the case where E contains a single element. Using the same averaging argument, $p(b, E)$ is smaller than the average of the K most profitable elements. Therefore, we can conclude that

$$P(S_{q'(S)}) \geq P(S) \cdot (1 - \frac{1}{K}) \cdot \left(1 - e^{-\frac{1}{\alpha}}\right).$$

■

By choosing a small enough α and a big enough K, the approximation ratio becomes $(\frac{e}{e-1} + \epsilon)$.

ALG3 is used by Algorithm 2 to find a $(1+\epsilon')$-approximate maximum profit selection that uses only bins in B and all the bins in B. This problem is identical to the Multiple Choice Knapsack Problem (MCKP) [5], defined as follows:

Instance: Disjoint classes $N_1, \ldots, N_m$ of items, and a capacity c. Each item $s \in N_j$ has a profit $p(s)$ and a weight $w(s)$.

Objective: Find a subset S' of items *with at most one item from each class* that has a feasible packing, such that the aggregated profit $\sum_{s_j \in S'} p(s_j)$ is maximized.

MCKP is known to have an FPTAS [5], which can be used as ALG3.

The running time of Algorithm 2 is $O((n \cdot m)^{2K})$ executions of greedy and $O(n^K)$ executions of ALG3, where m is the number of elements and n is the number of bins. Therefore, for every fixed K, the running time is polynomial in $n \cdot m$.

5 Conclusions and Extensions

We defined a new problem, referred to as the Generalized Maximum Coverage Problem (GMC), which is a generalization of the Maximum Coverage Problem (MC), the Budgeted Maximum Coverage Problem (BMC), and the Multiple Choice Knapsack Problem (MCKP). We also presented two algorithms for solving GMC. The first is a $(\frac{2e-1}{e-1} + \epsilon)$-approximation, and the second is an $(\frac{e}{e-1} + \epsilon)$-approximation with a much higher running time. Earlier works showed that MC is $\frac{e}{e-1} - \epsilon$ hard to approximate for every $\epsilon > 0$. Thus, our second algorithm is the best possible up to an ϵ.

GMC can be extended to the case where there are several budgets $L_1, L_2, \ldots, L_k$. Each budget has its own bins, but the elements are joint. In this extension, one needs to make a selection for

each budget, such that the credit for each element can be acquired at most once. In [2], the authors provide a framework that translates any α-approximation algorithm for maximizing the sum of profits for a single budget to an $(\alpha + 1)$-approximation algorithm for a similar problem but with multi-budgets. This framework, which is valid if for every feasible solution A any subset of A is also a feasible solution, can be used in order to get a $(\frac{2e-1}{e-1} + \epsilon)$-approximation, for every $\epsilon > 0$, to the this extension.

References

[1] A. A. Ageev and M. Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *IPCO*, pages 17–30, 1999.

[2] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Inform. Process. Lett.*, 100(4):162–166, 2006.

[3] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA '06: Proceedings of the Sixtieth annual ACM-SIAM symposium on Discrete algorithms, to appear*, 2006.

[4] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Co., Boston, MA, USA, 1997.

[5] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[6] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Inform. Process. Lett.*, 70(1):39–45, 1999.

[7] S. O. Krumke, M. V. Marathe, D. Poensgen, S. S. Ravi, and H.-C. Wirth. Budgeted maximum graph coverage. In *WG*, pages 321–332, 2002.

[8] H. Shachnai and T. Tamir. Polynomial time approximation schemes - a survey. Handbook of Approximation Algorithms and Metaheuristics. To appear. http://www.cs.technion.ac.il/∼hadas/PUB/ptas.pdf.

[9] D. B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474, 1993.