Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Heuristic and exact algorithms for Generalized Bin Covering Problem

S. Jabari [*1], D. Moazzami [†2] and A. Ghodousian [‡2]

[1]University of Tehran, Department of Algorithms and Computation.
[2]University of Tehran, College of Engineering, Faculty of Engineering Science

ABSTRACT

In this paper, we study the GENERALIZED BIN COVERING problem. For this problem an exact algorithm is introduced which can find optimal solution for small scale instances. To find a solution near optimal for large scale instances, a heuristic algorithm has been proposed. By computational experiments, the efficiency of the heuristic algorithm is assessed.

Keyword: Generalized Bin Covering Problem, heuristic algorithm, greedy algorithm

ARTICLE INFO

Article history:
Received 20, October 2015
Received in revised form 27, January 2016
Accepted 29, February 2016
Available online 18, March 2016

AMS subject Classification: Primary 05C78, 05C70

1 Introduction

In BIN COVERING problem there are m *bins* where each bin i has *revenue* (*profit*) r_i and *demand* d_i. Furthermore, there are n *indivisible items* where each item j has *size*

[*]Email:*sjabari@ut.ac.ir*
[†]E-mail: *dmoazzami@ut.ac.ir*
[‡]E-mail: *a.ghodousian@ut.ac.irr*

s_j. Bin i is *covered* or *filled* if the set of items assigned to it has total size at least d_i. In that case, the profit of bin is earned and the objective is to maximize the total profit. Bin Covering problem is a model to assign resource or task among some agents so that the goal is to maximize the number of agents who fulfill their quota [11]. This problem is dual of Bin Packing problem.

Bin Covering problem with respect to bins properties is categorized into three cases: classic, variable-sized and generalized. classic bin covering is the special case of the problem in which $d_j = r_j = 1$ for all bins. variable-sized bin covering is the special case of the problem in which $d_j = r_j$ for all bins. generalized bin covering is general mode of the problem so that d_j and r_j can get any value for all bins.

Moreover, there are two supply models for generalized and variable-sized bin covering problem: unit supply model and infinite supply model. In the unit model there is exactly one bin of each type, i. e., there are individual bins. By contrast, in the infinite supply model, there are arbitrarily many bins of each type. In this paper, we consider the offline Generalized Bin Covering problem in unit supply mode.

Bin Covering problem firstly was introduced by Assmann[2]. Assmann et al. presented efficient approximation algorithms for classic bin covering and they proved that Next Fit is a 2-approximation algorithm [1]. Also [4, 6, 13] presented asymptotic approximation algorithms for this problem and [5, 7, 8, 9, 12] worked on variable-sized bin covering. With the best of our knowledge, Hellwig worked on generalized bin covering and introduced a 5-approximation algorithm [12]. We call this algorithm "Hellwig algorithm" in this paper.

We consider an instance of Generalized Bin Covering: Suppose that $\beta = \{B_1, ..., B_m\}$ denotes a set of bins and let $\mu = \{I_1, ..., I_n\}$ be a set of items such that each item I_i has *size* s_i. Each bin B_j has two properties: *demand* d_j and *revenue* r_j. The objective is to pack items into as many bins as possible. If a bin is *covered*, then the revenue of the bin is gained. The goal of this problem is to gain maximum revenue.

Assmann showed by reduction of Partition problem that Bin Covering is actually an NP-hard problem. Since any approximation algorithm with ratio strictly smaller than 2 would have to solve the NP-complete Partition problem that is impossible (unless P=NP). So, there is no polynomial time $2 - \varepsilon$ approximation algorithm for this problem. On the other hand, there are certain applications that require exact solutions of NP-hard problems. In these cases, approximation algorithm for this problem may not be efficient. In section 2, an exact algorithm for generalized bin covering is introduced. In section

3, a heuristic algorithm has been proposed. To assess ability of the heuristic to solve the problem, we implemented Hellwig, heuristic and exact algorithms. The results of these algorithms for some instances of the problem are compared and showed in section 4.

2 Exact algorithm

Every NP-hard problems are solvable by enumerating all possible solutions. GENERALIZED BIN COVERING is a permutation problem and has a trivial $O(n^n)$ brute-force search (each item i, can be putted in m bins). The algorithm follows is an exponential exact and is not applicable in the real world but we know the design and analysis of exact algorithms leads to a better understanding of NP-hard problems and initiates interesting new combinatorial and algorithmic challenges[10].

We suppose (β, μ) as an instance of GENERALIZED BIN COVERING problem and we consider an optimal solution $OPT(\beta, \mu)$. For this instance, we create a weighted bipartite graph $G = (\beta \cup \mu, E)$ so that $E = \{ij | B_i \in \beta, I_j \in \mu : d_i \leq s_j\}$ and a weight function $\omega : E \to \mathcal{R}$ that $\omega_{ij} = r_i$ for all $ij \in E$. In this graph, we find maximum weighted matching. Hungarian algorithm [14] can find maximum weighted matching in bipartite graph. The matching finds a solution for the GENERALIZED BIN COVERING problem. This solution is at least as good as the part of the optimal solution that bins are covered by one item [12].

Without loss of generality, we can assume in OPT, all bins are covered by one item since when a bin is covered by more than one item, all these items are assumed as one item.

Each solution is an assignment of items to bins. The exact algorithm finds the partition of items that occurs in an optimal solution. For this purpose, the algorithm must consider all possible partitions on set of items. Suppose the number of all possible partitions on set of items be P. In each partition $part_k$ $1 \leq k \leq P$, the exact algorithm does the following: $part_k = \{p_1, \cdots, p_l\}$. Each subset p_i, $1 \leq i \leq l$ is considered as one item. The size of p_i is $\sum_{I_q \in p_i} s_q$. Algorithm creates a weighted bipartite graph by the bins and $\{p_1, \cdots, p_l\}$ as items. Then finds a solution for $part_k$ by finding maximum weighted matching in bipartite graph. After checking all possible partitions, the partition(s) with the most revenue is (are) optimal. This exact algorithm described in Figure 1.

Complexity Items can be partitioned at most to $min\{m, n\}$ subsets. So if $m < n$ then $P = \sum_{i=1}^{m} S(n, i)$ (Stirling number of the second kind). Or else $P = B(n)$ (Bell number). We know $\sum_{i=1}^{n} S(n, i) \leq n!$. In the process of the exact algorithm, we must run

Figure 1: Exact algorithm

```
1. $p = min\{m, n\}$
2. $partitions$ =all partitions of set of items from one subset
   to $p$ subsets.
3. $partNumber = \sum_{i=1}^{p} S(n, i)$
4. For each $partitions_i$ do
   $i = 1, \ldots, partNumber$
   (a) $k$= Number of subsets in $partitions_i$.
   (b) For each subset $S_j$ of $partitions_i$
       $j = 1, \ldots, k$
       -Merge the items of $S_j$ to create one item.
   (c) Create a weighted bipartite graph $G_i$ by bins and $k$
       new items.
   (d) Find maximum weighted matching for $G_i$ by Hungar-
       ian algorithm as $solution_i$.
5. Select solution with the most revenue from $solution$ array
   as $optimal\ solution$.
```

Hungarian algorithm for each partition. Since Hungarian algorithm has n^3 running time, the complexity of the exact algorithm is $O(n^3 n!)$ that is equivalence to $O^*(n!)$. Another interesting question is on the space requirements of the algorithm. This algorithms does not need exponential space.

3 Heuristic algorithm

This heuristic algorithm is a combinatorial algorithm and contains four algorithms: Maximum Weighted Matching in bipartite graph and three greedy algorithms. As mentioned, Maximum Weighted Matching in bipartite graph give a solution that is at least as good as the part of the optimal solution that bins are covered by one item. So, if most of revenue in optimal solution achieved from the bins that are covered by one, this solution can be reasonable. Otherwise we use greedy algorithms.

A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum [3]. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time. We design three greedy algorithms. For these greedy algorithms, *efficiency* of bins are important besides their demand and revenue. For bin B_i the efficiency, e_i, is equal to r_i/d_i. The greedy algorithms apply two priority queues to order bins: *efficiency* queue and *revenue* queue. Efficiency queue sorts the bins non-increasingly by efficiency. If two bins have same efficiency, the bin with more revenue has more priority. Revenue queue sorts the bins non-increasingly by revenue. If two bins have same revenue, then the bin with more efficiency has more priority.

Efficiency algorithm is one of the greedy algorithms. This algorithm selects the bins from efficiency queue. Another is *revenue algorithm* which selects the bins form revenue queue. The last one is called *fair algorithm* that selects the bins fairly. One bin from revenue queue and the next from efficiency queue or conversely. These algorithms are described as follow:

1. **Efficiency algorithm:** The bins are selected from efficiency queue and items are sorted non-increasingly. This algorithm tries to cover each bin in three stages. In each stage, if needed items are found then the algorithm covers the bin and selects next bin to cover. Also, if the algorithm can't cover the bin it begins to cover the next bin. These stages are:

   (a) The algorithm finds an item equal to the demand of the bin.

   (b) It searches items smaller than the demand of the bin. If the sum of these items isn't smaller than its demand, the bin is covered.

   (c) It finds smallest item that is bigger than the demand of the bin.

   When an item is assigned, it will be omitted from the list of items. This algorithm is described in Figure 2.

2. **Revenue algorithm:** This algorithm is similar to efficiency algorithm and there are only one difference so that bins are selected from revenue queue.

3. **Fair algorithm:** This algorithm selects randomly the first bin from revenue queue or efficiency queue. When a queue turned out, the next bin is selected from other queue. By selecting a bin, the algorithm searches items to find one item whose size is at least equal to the demand of the bin. If there exists such this item then the bin is covered. Otherwise, the algorithm searches to find items whose sizes are less

Figure 2: Efficiency algorithm

```
(a) Sort items non-increasingly.
(b) Select bins from efficiency queue.
(c) For each $B_i$ do
    $i = 1, \ldots, m$
      i. Find the first item $I_j$ so that $s_j \geq d_i$.
     ii. If there isn't such item then $j = n + 1$.
    iii. If $s_j = d_i$ then assign $I_j$ to $B_i$.
         Else, if $\Sigma_{k=1}^{j-1} s_k \geq d_i$ then
           A. Find $l$ so that $\Sigma_{k=1}^{l} s_k \geq d_i$ and $\Sigma_{k=1}^{l-1} s_k < d_i$.
           B. Assign items $I_1, \ldots, I_l$ to $B_i$
         Else, if $j \leq n$ then assign $I_j$ to $B_i$ .
     iv. Remove the assigned items from the list of items.
```

than the demand of the bin. The bin is covered, if sum of these items are at least equal to the bin demand.

Algorithm Analysis The heuristic algorithm solves the Generalized Bin Covering problem by four methods (maximum matching and greedy algorithms) to find a reasonable solution. Each algorithm finds a solution for the problem. The solution with the best result is introduced as the heuristic solution.
Greedy algorithms have $O(n^2)$ time complexity. So the running time of the heuristic algorithm is dominated by the complexity of Hungarian algorithm so is $O(n^3)$.

4 Computational Experiments

In this section we assess the efficiency of the heuristic algorithm. To get this aim, this solution is compared with the optimal (if possible) or Hellwig solution. For this comparison, a database is created. This database includes random instances of the problem. Bins and items properties are selected randomly from their domains. These domain can be changed. Database contains two sets of problems: in the first set, there are small instances (at most 12 bins and items) so that we can gain optimal solution for them. The second set contains almost large instances. In the most of instances, the number of items are more than bins and the size of items are reduced than the demand of bins. This property of instances constrains their optimal solutions so that more covered bins are filled by more than one item. For this assessment, this property is important because finding a good solution is difficult when a large fraction of bins covered by several bins in the optimal solution (Hungarian algorithm can find the part of optimal solution that bins are covered by one item).

Table 1: The ratio of the optimal revenue to the heuristic or Hellwig revenue (α)

bins number	items number	instances number	α	heuristic	Hellwig
2-6	3-7	10,000	$\alpha = 1$	88.45%	62.07%
			$1 < \alpha \leq 1.5$	10.2%	28.56%
			$1.5 < \alpha \leq 2$	0.33%	7.77%
			$2 < \alpha \leq 2.5$	0%	0.24%
			$2.5 < \alpha \leq 3$	0%	0.02%
6-8	7-10	10,000	$\alpha = 1$	63.05%	24.46%
			$1 < \alpha \leq 1.5$	36.73%	69.42%
			$1.5 < \alpha \leq 2$	0.22%	5.8%
			$2 < \alpha \leq 2.5$	0%	0.29%
			$2.5 < \alpha \leq 3$	0%	0.03%
8-10	10-12	1000	$\alpha = 1$	39.6%	9.8%
			$1 < \alpha \leq 1.5$	60.4%	86.5%
			$1.5 < \alpha \leq 2$	0%	3.4%
			$2 < \alpha \leq 2.5$	0%	0.3%
			$2.5 < \alpha \leq 3$	0%	0%

Table 2: Comparison the revenue of the heuristic with the revenue of Hellwig (small instances)

bins number	items number	more revenue		equal revenue
		heuristic	Hellwig	
2-6	3-7	31.28%	0.6%	68.11%
6-8	7-10	61.96%	1.97%	36.07%
8-10	10-12	73.8	0.21%	24.1%

For small instances of the problem, we calculated the heuristic, Hellwig and optimal solutions. α is ratio of the optimal revenue to the heuristic or Hellwig revenue and for each instance is calculated. Table 1 shows the results of this comparison. α isn't more than 3 in non of instances. The results demonstrate that heuristic algorithm can find the

optimal solution in more cases and if can't find the optimal solution, it ables to find a solution near to the optimal.
In table 2 Hellwig and the heuristic solutions compared to each other (database is the same as database of table 1). This table show that the heuristic algorithm has better solutions for larger instances in comparison with Hellwig algorithm.

For almost large instances, only Hellwig and heuristic solutions calculated and they compared to each other. The result is in table 3. Each row of table contains $10,000$ problem instances. Results show that for most of the instances, the heuristic solution is better than Hellwig solution and for larger instances the heuristic solution give us better results.
The consumed time of these algorithms is in table 4. Heuristic runs faster than Hellwig algorithm.

Table 3: The revenue of the heuristic with the revenue of Hellwig (almost large instances)

bins number	items number	more revenue		equal revenue
		heuristic	Hellwig	
20 – 40	30 – 50	99.18%	0.05%	0.77%
50 – 80	50 – 100	95.82%	0.04%	4.14%
100 – 150	100 – 200	97.06%	0%	2.94%
150 – 200	200 – 300	99.99%	0.01%	0%

Table 4: Consumed time of Hellwig and heuristic for almost large instances

bins number	items number	heuristic	Hellwig
20 – 40	30 – 50	35s	57s
50 – 80	50 – 100	107s	169s
100 – 150	100 – 200	298s	478s
150 – 200	200 – 300	686s	1,128s

5 Conclusion

In this paper we have studied off-line Generalized Bin Covering Problem in unit supply model. First, we have introduced an exact algorithm with $O^*(n!)$ time complexity. This is not applicable for large problem instances. For larger instances, we have proposed a heuristic algorithm. In computational experiments, the algorithm was compared to the exact and Hellwig algorithms. The results show that the heuristic algorithm can find the optimal solution or a solution near to optimal in most cases. Also, almost in the most of cases the heuristic solution is better than the Hellwig solution specially for larger problem instances.

6 Acknowledgment

This work was supported by Tehran University. Our special thanks go to the University of Tehran, College of Engineering and Department of Engineering Science for providing all the necessary facilities available to us for successfully conducting this research. The first author is thankful to Ali Golshani of Department of Algorithms and Computation, University of Tehran, Iran for his valuable comments and suggestions.

References

[1] S. F. Assman, D. Johnson, D. J. Kleitman, and J.-T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5(4):502–525, 1984.

[2] F. S. Assmann. Problems in discrete and applied mathematics. 1983.

[3] P. E. Black. *Dictionary of Algorithms and Data Structures.* National Institute of Standards and Technology, 2005.

[4] J. Csirik, J. B. Frenk, M. Labbe, and S. Zhang. Two simple algorithms for bin covering. *Acta Cybernetica*, (14):13–25, 1999.

[5] J. Csirik and J. B. G. Frenk. A dual version of bin packing. *Algorithms Review*, 2:87–95, 1990.

[6] J. Csirik, D. S. Johnson, and C. Kenyon. Better approximation algorithms for bin covering. *12th Symposium on Discrete Algorithms*, pages 557–566, 2001.

[7] J. Csirik and V. Totik. Online algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, (21):163–167, 1988.

[8] J. Csirik and G. J. Woeginger. On-line packing and covering problems. *In Online Algorithms*, 1442:147–177, 1998.

[9] L. Epstein. Online variable sized covering. *Information and Computation*, 171(2):294–305, 2001.

[10] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms.* Springer, 2010.

[11] A. S. Fukunaga and R. E. Korf. Bin-completion algorithms for multicontainer packing and covering problems. *Journal of Artificial Intelligence Research*, (28):393–429, 2007.

[12] M. Hellwig and A. Souza. Approximation algorithms for generalized and variable-sized bin covering. In L. N. in Computer Science, editor, *15th International Workshop, Approximation, Randomization, and Combinatorial Optimization*, volume 7408, pages 195–204. Springer Berlin Heidelberg, 2012.

[13] K. Jansen and R. Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoretical Computer Science*, 306(1–3):543– 551, 2003.

[14] H. W. Kuhn. *50 Years of Integer Programming 1958-2008*, chapter The Hungarian Method for the Assignment Problem, pages 7–28. Springer Berlin Heidelberg, 2010.