Open-End Bin Packing Problem with Conflicts

Ece Nur Balık, Ali Ekici, and Okan Örsan Özener

Ozyegin University, Department of Industrial Engineering,
Nisantepe Mah. Cekmekoy 34794 Istanbul, Turkey
`ece.balik@ozu.edu.tr`

Abstract. In this paper, we study the *Open-End Bin Packing Problem with Conflicts* (OEBPPC) which is a combination of two variants of the famous bin packing problem: *Open-End Bin Packing Problem* and *Bin Packing Problem with Conflicts.* In OEBPPC, the aim is to pack all of the items into minimum number of bins where the bin capacity is allowed to be exceeded only by the last item placed and there exist conflicts between some item pairs. We introduce a mathematical formulation for the problem and adapted some known heuristics and a metaheuristic algorithm to this new problem.

Keywords: bin packing problem with conflicts, open-end bin packing, heuristic, metaheuristic

1 Introduction

Bin Packing Problem (BPP) is a well-known combinatorial problem that aims to pack a set of items into fixed sized bins, using as few bins as possible. Gendreau et al. introduces a variant of BPP known as the *Bin Packing Problem with Conflicts* (BPPC) in [3]. In BPPC, certain pairs of items are in conflict, and such items cannot be packed into the same bin. Another variant of BPP, called the *Open-End Bin Packing Problem* (OEBPP), is introduced by Leung et al. in [4]. In OEBPP, the bin capacity can be exceeded, only by the last item (called the *overflow item*) placed into that bin.

In this study, we present a new variant, called the *Open-End Bin Packing Problem with Conflicts* (OEBPPC), which is the combination of BPPC and OEBPP. In OEBPPC, conflicting items cannot be placed into the same bin, and the bin capacity can be exceeded by the last item placed into the bin. The objective is to pack a set of items into a minimum number of bins. To the best of our knowledge, OEBPPC has not been studied in the literature. We develop adaptations of heuristic algorithms proposed for BPPC and a metaheuristic proposed for BPP to address OEBPPC.

2 Problem definition and a mathematical model

We use $V = \{1, 2, ...i, ..., n\}$ to denote the set of items where item i has weight w_i. Conflicts between item pairs are represented with a conflict graph $G = (V, E)$

where an existing edge $(i, j) \in E$ implies a conflict between items i and j. The capacity of a bin is denoted by C.

For $i, j \in V$, we define the following decision variables to develop an integer program for OEBPPC:

$$x_{ij} = \begin{cases} 1, \text{ if item } i \text{ is assigned to the bin in which the overflow item is item } j \\ 0, \text{ otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1, \text{ if item } i \text{ is the overflow item in its bin} \\ 0, \text{ otherwise.} \end{cases}$$

$$\text{Min} \quad \sum_{i \in V} y_i \tag{1}$$

s.t.

$$y_i + \sum_{i \neq j \in V} x_{ij} = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{i \neq j \in V} w_i x_{ij} \leq (C-1) y_j \quad \forall j \in V \tag{3}$$

$$x_{ij} + y_j \leq 1 \quad \forall (i,j) \in E,\ i \neq j \tag{4}$$

$$x_{ik} + x_{jk} \leq 1 \quad \forall (i,j) \in E, k \in V,\ i \neq j \neq k \tag{5}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in V \tag{6}$$

$$y_j \in \{0,1\} \quad \forall j \in V \tag{7}$$

The objective (1) is to minimize the number of *overflow items*, hence the number of bins used. Constraints (2) ensure that each item is assigned to a bin, either being an overflow item of that bin or not. Constraints (3) guarantee that the overall weight of items assigned to a bin (except the *overflow item*) must leave at least one-unit capacity available for the *overflow item.* Conflicting items cannot be packed into the same bin with the help of constraints (4) and (5).

3 Algorithms Tested

To address OEBPPC, we adapted two types of algorithms from the literature:

1. Muritiba et al. modify the famous FFD, BFD and WFD algorithms for BPPC using *surrogate weights* [5]. The *surrogate weight* of an item is calculated as a combination of the number of conflicts it has with other items and its weight. These algorithms are denoted by FFD(α), BFD(α) and WFD(α) where $\alpha \in \{0, 0.1, 0.2, ..., 0.9, 1\}$ is the parameter used to calculate surrogate weights. Each algorithm is run for different parameter values, and the best result is reported.

2. Falkenauer introduces the algorithm called the *Grouping Genetic Algorithm* (GGA) to address BPP [1]. In GGA, each gene represents a group of items that are packed to a bin. This structure helps to preserve well-filled bins through generations. By using several operators, he concludes that GGA performs better than FFD especially when the residual capacity of bins are smaller.

We modified FFD(α), BFD(α), WFD(α) and GGA to address OEBPPC. To test the performance of the algorithms, we used the first class of BPP instances (u120) from [2]. In that class, we used the first 10 base instances with 120 items. Item weights are uniformly distributed as integers between 20 and 100, and the bin capacity is 150. For each of these instances, we randomly generated conflict graphs of specific density (δ) values ($\{0.0, 0.1, 0.2, ..., 0.9\}$) and obtained 100 instances. We terminate GGA after having 100 consecutive iterations without improving the number of bins in the best solution. The average optimality gaps (OG) of the solutions and the average CPU times (in seconds) of the algorithms for each δ value are presented in Table 1. Optimality gaps are calculated using the lower bounds obtained by solving the integer model with CPLEX. We observe that GGA outperforms FFD(α), BFD(α) and WFD(α) in terms of solution quality. However, it requires significantly more CPU time.

Table 1. Comparison of algorithms on 100 instances with 120 items

δ	**FFD(α)**		**BFD(α)**		**WFD(α)**		**GGA**	
	OG	**CPU**	**OG**	**CPU**	**OG**	**CPU**	**OG**	**CPU**
0	10.3%	0.002	10.3%	0.003	10.3%	0.008	3.3%	2483.2
0.1	10.3%	0.003	10.3%	0.005	11.0%	0.006	4.0%	1842.3
0.2	11.3%	0.003	11.3%	0.005	12.3%	0.006	4.6%	2178.5
0.3	12.9%	0.002	12.9%	0.006	12.9%	0.009	5.6%	1771.5
0.4	14.9%	0.006	15.2%	0.006	14.9%	0.006	6.6%	1754.7
0.5	17.2%	0.005	18.2%	0.006	18.9%	0.006	6.3%	2679.0
0.6	21.2%	0.002	21.6%	0.003	22.5%	0.011	9.6%	4026.0
0.7	28.1%	0.003	27.1%	0.008	30.4%	0.008	12.9%	4465.8
0.8	27.1%	0.005	26.5%	0.009	32.8%	0.005	12.3%	5017.3
0.9	22.3%	0.006	22.3%	0.009	26.5%	0.012	6.6%	4015.2
Average	**17.6%**	**0.004**	**17.6%**	**0.006**	**19.3%**	**0.008**	**7.2%**	**3023.3**

References

1. Falkenauer, E.: A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems. Evolutionary Computation, vol.2, pp. 123-144, 1994. doi:10.1162/evco.1994.2.2.123
2. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics, vol.2, pp.5-30, 1996. doi:10.1007/BF00226291
3. Gendreau, M., Laporte, G., Semet, F.: Heuristics and lower bounds for the bin packing problem with conflicts. Computers & Operations Research vol. 31, pp. 347-358, 2004. doi:10.1016/S0305-0548(02)00195-8
4. Leung, J., Dror, M., Young, G.: Technical note: A note on an open-end bin packing problem. Journal of Scheduling vol. 4, pp. 201-207, 2001. doi:10.1002/JOS.75
5. Muritiba, A. E. F., Iori, M., Malaguti, E., Toth, P.: Algorithms for the bin packing problem with conflicts. INFORMS Journal on Computing vol. 22, pp. 401-415, 2010. doi:10.1287/ijoc.1090.0355