The Generalized Bin Packing Problem: Models and Bounds

Teodor Gabriel Crainic
CIRRELT, Montreal, Canada

Roberto Tadei, Guido Perboli, Mauro Maria Baldi,
Department of Computer and Control Engineering,
Politecnico di Torino, Turin, Italy
Email: mauro.baldi@polito.it

1 Introduction

We present the Generalized Bin Packing Problem (GBPP), a new packing problem where, given a set of items characterized by volume and profit and a set of bins with given volumes and costs, one aims to select the subsets of *profitable* items and appropriate bins to optimize an objective function which combines the cost of using the bins and the profit yielded by loading the selected items. The GBPP thus aims to generalize many other packing problems such as the Bin Packing Problem (BPP), the Variable Size Bin Packing Problem (VSBPP), the Variable Cost and Size Bin Packing Problem (VCSBPP), the Knapsack Problem (KP), and the Multiple Knapsack Problem (MKP). It also provides the means to identify and analyze the trade-off between item and bin selections that are part of many transportation and logistics planning problems [1].

From a practical point of view, the GBPP models many real-life situations, including scenarios where orders may be urgent or a dispatching priority is associated to each item to be shipped. Moreover this paper contributes to enhance the transportation literature and to cope some issues in the specific literature on packing problems [2, 3].

We present two mixed integer programming formulations of the GBPP, as well as lower and upper bounds. The efficiency and accuracy of these bounds are proved by means of an extensive set of computational results.

2 The GBPP and its formulations

The GBPP considers a set of items characterized by volume and profit and sets of bins of various types characterized by volume and cost. Part of the items, denoted as *compulsory*, must be loaded, while a selection has to be made among the *non-compulsory* ones. The objective is to determine which non-compulsory items should be taken and select the bins to load the compulsory and the taken non-compulsory items to minimize the total net cost, computed as the difference between the total cost of the used bins and the total profit of the loaded items.

Let $\mathcal{I}$ denote the set of items, with $n = |\mathcal{I}|$, and w_i and p_i be the volume and the profit of item $i \in \mathcal{I}$, respectively. Define $\mathcal{I}^C \subseteq \mathcal{I}$ the subset of items that are forced to be loaded, and $\mathcal{I}^{NC} = \mathcal{I} \setminus \mathcal{I}^C$ the subset of non-compulsory items. Let $\mathcal{J}$ denote the set of available bins, with $m = |\mathcal{J}|$ and $\mathcal{T}$ be the set of bin types. For any bin $j \in \mathcal{J}$, let $\sigma(j) \in \mathcal{T}$ be the type of bin j. Define for each type $t \in \mathcal{T}$, the minimum L_t and the maximum U_t number of bins of that type that may be selected, as well as the selection cost C_t and volume W_t of a bin of type $t \in \mathcal{T}$. Finally, let us denote $U \leq \sum_{t \in \mathcal{T}} U_t$ the maximum number of bins of any type one can use. The item-to-bin accommodation rules of the GBPP may then be stated as follows: all items in I^C must be loaded; for all used bins, the sum of the volumes of the items loaded into a bin must be less than or equal to the volume of bin itself; the number of bins used for each type $t \in \mathcal{T}$ must be within the lower and upper bounds L_t and U_t; the total number of used bins cannot exceed U.

A first model of the GBPP can be formulated as an assignment model, which involves selection binary variables y_j equal to 1 if bin $j \in \mathcal{J}$ is used, 0 otherwise and item-to-bin assignment binary variables x_{ij} equal to 1 if item $i \in \mathcal{I}$ is loaded into bin $j \in \mathcal{J}$, 0 otherwise. This model, named AM, involves a polynomial number of variables and constraints.

The second model is based on a set covering formulation related to the feasible loading patterns for the bin types. A *feasible loading pattern* for a bin of type $t \in \mathcal{T}$ corresponds to a set of items that may be loaded into the bin while respecting all dimension restrictions and accommodation rules. We name SC this second formulation. Involving an exponential number of variables, this model is specially suited for column-generation based methods.

3 Bounds

The two models introduced in Section 2 are the starting point to get two lower bounds, one for each model, and the best among the two will be taken. Upper bounds can also be computed via constructive heuristics or by exploiting the SC model. In the following subsections we present the methods for calculating lower and upper bounds.

3.1 Lower bounds

Following [3], a first lower bound, named LB_1, can be obtained by aggregating some constraints of the AM model. The resulting problem is an aggregate Knapsack Problem which, solved to optimality, yields the lower bound LB_1. The second lower bound, named LB_2, is computed from the linear relaxation of the SC model through column generation, which provides the means to implicitly deal with the large number of variables in the model (and it is widely used in packing problems [4, 5]). The general column generation approach when applied to the GBPP consists in starting with a relatively small set of feasible patterns $\mathcal{P}$ corresponding to a restricted GBPP, named R-GBPP. The scope is to implicitly enumerate the exponential number of patterns involved (see [6] for further details).

3.2 Upper bounds

Upper bounds can be computed following two different approaches: 1) via constructive heuristics or 2) by exploiting the patterns produced by the column generation procedure. The constructive heuristics we considered are the well known *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics [7]. They are widely used for the BPP and we adapted them to the GBPP.

The second approach considers the patterns produced by the column generation procedure. A first upper bound is obtained by exactly solving the SC model by using a Branch-&-Bound which only considers the columns generated by the column-generation procedure while computing the lower bound. This may still be quite time consuming, however. Consequently, we stop with the Branch-&-Bound after a given computation time and we name Z_{SC} the resulting value of the objective function which is an upper bound to the GBPP. A second upper bound is based on *diving*, a well-known method for finding good quality integer solutions from continuous relaxations. The working principle is to iteratively round up variables and re-optimize the continuous relaxation.

4 Computational Results

Since there are no instances in literature for the GBPP, we started from the instance set of Monaci [2] introduced for the VSBPP. We name Class 0 this instance set. It consists in 300 instances where all the items are compulsory. We then created two more instance sets, named Class 1 and Class 2, which are like Class 0 but with the addition of the item profits and where all the items are non-compulsory.

Table 1 shows percentage upper bound gaps computed with respect to our best lower bound. Note that, since Class 0 is made up by the instances of Monaci, the gaps of this set can be computed with respect to the Monaci optima, when available. Otherwise they are computed with respect to our best lower bound. Column 1 shows the instance set, Column 2 the Z_{SC} gap, Columns 3 and 4 two diving gaps computed by following two diving strategies, Columns 5 and 6 the best among the previous diving strategies and

BFD. Finally Column 7 shows BFD results.

CLASS	$\mathbf{Z_{SC}}$	**D1**	**D2**	**B-D1**	**B-D2**	**BFD**
0	0.14	1.30	0.98	0.65	0.62	1.62
1	0.10	0.95	1.37	0.51	0.55	2.30
2	0.07	0.85	1.32	0.42	0.49	1.81
% MEAN	**0.11**	**1.04**	**1.22**	**0.53**	**0.55**	**1.91**

Table 1: Upper bound comparisons

The best results are yielded by Z_{SC} which is, however, the most time consuming method. Well balanced results, in terms of efficiency and accuracy, are yielded by the diving methods (D1 and D2) and the diving integrated with BFD methods (B-D1 and B-D2). The BFD heuristic has a negligible computational time but it yields the worst gaps. Further results will be provided during the conference.

References

[1] A. Imai, E. Nishimura, S. Papadimitriou, K. Shintani, "The container shipping network design problem with empty container repositioning", *Transportation Research Part E*, 43(1), 39–59, 2007.

[2] M. Monaci, *Algorithms for packing and scheduling problems*, PhD thesis, Università degli Studi di Bologna, Bologna, Italy, 2001.

[3] T. G. Crainic, G. Perboli, W. Rei, and R. Tadei, "Efficient lower bounds and heuristics for the variable cost and size bin packing problem", *Computers and Operations Research*, 38, 1474–1482, 2011.

[4] C. Alves and J.M. Valerio De Carvalho, "Accelerating column generation for variable sized bin-packing problems", *European Journal of Operational Research*, 183, 1333–1352, 2007.

[5] F. Vanderbeck, "Computational study of a column generation algorithm for bin packing and cutting stock problems", *Mathematical Programming*, 86, 565–594, 1996.

[6] M. M. Baldi, T. G. Crainic, G. Perboli, and R. Tadei. "The generalized bin packing problem", *CIRRELT*, CIRRELT-2011-39, 2011.

[7] S. Martello and P. Toth, *Knapsack Problems - Algorithms and computer implementations*, John Wiley & Sons, Chichester, UK, 1990.