POLITECNICO DI TORINO
Repository ISTITUZIONALE

The Generalized Bin Packing Problem

Original
The Generalized Bin Packing Problem / Baldi, M.M., Crainic, T., Perboli, G., Tadei, R.. - In: TRANSPORTATION RESEARCH PART E-LOGISTICS AND TRANSPORTATION REVIEW. - ISSN 1366-5545. - STAMPA. - 48:(2012), pp. 1205-1220. [10.1016/j.tre.2012.06.005]

Availability:
This version is available at: 11583/2293548 since: 2017-12-04T19:45:47Z

Publisher:
ELSEVIER

Published
DOI:10.1016/j.tre.2012.06.005

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

06 November 2026

The Generalized Bin Packing Problem

Mauro Maria Baldi[a,*], Teodor Gabriel Crainic[c,b], Guido Perboli[a,b], Roberto Tadei[a]

[a] *Politecnico di Torino, Turin, Italy*
[b] *CIRRELT, Montreal, Canada*
[c] *School of Management, UQAM, Montreal, Canada*

Abstract

In the Generalized Bin Packing Problem (GBPP), given two sets of compulsory and non-compulsory items characterized by volume and profit and a set of bins with given volume and cost, we want to select the subset of profitable non-compulsory items to be loaded together with the compulsory ones into the appropriate bins in order to minimize the total net cost. Lower and upper bounds to the GBPP are given. The results of extensive computational experiments show that the proposed procedures are efficient and the bounds are tight.

Keywords: Generalized Bin Packing, Heuristics, Column Generation, Bounds.

1. Introduction

In this paper, we introduce the Generalized Bin Packing Problem (GBPP). In this new bin packing problem, given two sets of compulsory and non-compulsory items characterized by volume and profit and a set of bins with given volume and cost, we want to select the subset of profitable non-compulsory items to be loaded together with the compulsory ones into the appropriate bins in order to minimize the total net cost. The total net cost is given by the difference between the total cost of the selected bins and the total profit of the loaded items.

We refer to the GBPP to address problems in the field of logistics, where changes arising in the supply chain and fleet management due to cross-continental fleet flows and multi-modal and green logistics have forced researchers and practitioners to redefine their processes [3, 21]. Our paper is a contribution in this direction, as it defines a packing problem that is able to simultaneously consider bin costs and item profits and takes into account restrictions on the bin availability and their heterogeneity in terms of cost and volume. The GBPP also brings innovation in the area of airfreight transportation, where items are loaded according to their volume [15]. Here the GBPP is able to describe the fundamental role played by the trade-off between shipping costs and item profits, which arises in all transportation settings.

The GBPP generalizes the following problems addressed by the bin packing literature: the Bin Packing Problem (BPP) [16], the Variable Size Bin Packing Problem (VSBPP) and the Variable Cost and Size Bin Packing Problem (VCSBPP) [4, 17, 5]. Whilst in the BPP all bins have the same capacity, in the VSBPP they have different capacities and in the VCSBPP they differ in both capacity and cost. The GBPP also generalizes the following knapsack problems: the Knapsack Problem (KP), the Multiple Knapsack Problem (MKP) and the Multiple Knapsack Problem with identical capacities (MKPI) [16].

We present two mixed integer programming formulations of the GBPP. The first is based on item-to-bin assignment decisions and requires a polynomial number of variables and constraints. This formulation is useful to discuss how the GBPP generalizes the packing and knapsack problems mentioned above, but it is not computationally efficient. It is, however, the starting point for computing our first lower bound to the GBPP. We thus introduce a second

*Corresponding author.
Department of Control and Computer Engineering
Tel. +39 011 0907083
E-mail: mauro.baldi@polito.it

model based on feasible loading patterns and set covering ideas. Despite requiring an exponential number of variables, this formulation is much more efficient. We thus present several procedures to compute lower and upper bounds to the GBPP and show their accuracy and efficiency through extensive computational experiments. A large number of instance sets for the GBPP are introduced. The instance sets are designed to challenge the proposed procedures and thus provide insight into the impact of different parameters on the optima.

The paper is organized as follows: Section 2 gives a literature review of the problems the GBPP generalizes. The two GBPP formulations are introduced in Section 3, whilst lower and upper bounds are presented in Sections 4 and 5, respectively. Instance sets and computational results are presented and discussed in Section 6. Finally, in Section 7 our conclusions are drawn.

2. Literature Review

The problem settings most immediately generalized by the GBPP are the BPP and the VCSBPP. Here we briefly recall the literature related to these problems.

The objective of the BPP is to load all items while minimizing the number of used bins. The problem has been extensively studied in the past decades, producing several exact and heuristic methods [16]. The first bounds were proposed by Martello and Toth [16]. New lower bounds were developed by Fekete and Schepers [9] by means of dual feasible functions. On the basis of this paper, Crainic et al. [7, 6] developed fast and more accurate lower bounds, able to reduce the optimality gap for a number of hard instances. A different approach was defined in [22], where the author proposed a formulation with an exponential number of variables and a column generation lower bound procedure for the Bin Packing and the Cutting Stock problems. Several heuristics were also proposed, e.g., the polynomial-time approximation schemes of de la Vega and Lueker [8] and Karmarkar and Karp [13] allowing to approximate an optimal solution within $1 + \epsilon$, for any fixed $\epsilon > 0$. However, these results are difficult to apply in practice, due to the enormous size of the constants characterizing the polynomials. The literature shows that the most commonly used methods to achieve computing efficiency together with solution quality are the *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics, sometimes combined with local improvement heuristics [19].

The VCSBPP is a generalization of the BPP, where all items must be loaded, but bins can be chosen among several types differing in volume and cost. The total accommodation cost, computed as the total cost of the used bins, must be minimized. A number of studies have recently been dedicated to the VCSBPP. Correia et al. [4] proposed a formulation that explicitly included the bin volumes occupied by the corresponding packings, together with a series of valid inequalities improving the quality of the lower bounds obtained from the linear relaxation of the proposed model. The authors also introduced a large set of instances with up to 1000 items and used them to analyze the quality of the lower bounds. Crainic et al. [5] proposed tight lower and upper bounds, which can be computed within a very limited computing time, and were able to solve to optimality all the instances proposed in [4]. The authors also presented a first computational study of the sensitivity of the optimal cost with respect to the cost definition [5].

A special case of the VCSBPP is the Variable Size Bin Packing Problem (VSBPP) where bin costs are equal to their associated volumes. The aim of the VSBPP is then to minimize the wasted volume. Monaci [17] presented a series of lower bounds and solution methods (both heuristic and exact) for the VSBPP. The author also introduced instance sets for the problem considering up to 500 items. His exact method was able to solve most instances to optimality. Kang and Park [12] developed two greedy algorithms for another special case of the VSBPP, where the unit cost of each bin does not increase as the bin volume increases, and analyzed their performances on instances with and without divisibility constraints. Seiden et al. [20] proposed upper and lower bounds for the on-line version of the problem.

As we mentioned in the introduction, the GBPP is also able to generalize three knapsack problems: the KP, the MKP, and the MKPI. One bin only, called the knapsack, is given in the KP and the goal is to select appropriate items in order to maximize the overall profit while satisfying capacity constraints. The MKP is a generalization of the KP due to the presence of more than one knapsack. Finally, the MKPI is a particular case of the MKP, where all the knapsacks have the same capacity. All these problems are thoroughly discussed in [16, 18, 14].

3. Problem Definition and Formulation

The GBPP considers a set of items characterized by volume and profit and sets of bins of various types characterized by volume and cost. A subset of the items which we call *compulsory* must be loaded, while a selection has to be made among the *non-compulsory* ones. The objective is to select the non-compulsory items to load and the bins into which to load the compulsory and the selected non-compulsory items in order to minimize the total net cost. This is given by the difference between the total cost of the used bins and the total profit of the loaded items.

We start this section with a formal description of the GBPP and then we introduce two formulations of the problem. The first formulation extends to the GBPP the assignment model of the BPP [16]. Even though this type of formulation is not often used in practice, we exploit it to derive a first lower bound. Then, we consider a set covering formulation of the problem, which is the starting point for a column generation procedure able to derive a second lower bound. Moreover, by exploiting the patterns created through the column generation procedure, accurate upper bounds can also be computed.

3.1. Notation

Let $\mathcal{I}$ denote the set of items and w_i and p_i be the volume and profit of item $i \in \mathcal{I}$. Define $\mathcal{I}^{\mathrm{C}} \subseteq \mathcal{I}$ the subset of compulsory items and $\mathcal{I}^{\mathrm{NC}} = \mathcal{I} \setminus \mathcal{I}^{\mathrm{C}}$ the subset of non-compulsory items. Let $\mathcal{J}$ denote the set of available bins and $\mathcal{T}$ be the set of bin types. For any bin $j \in \mathcal{J}$, let $\sigma(j) = t \in \mathcal{T}$ be the type t of bin j. Define for each bin 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 cost C_t and the volume W_t of the bin. Finally, denote $U \leq \sum_{t \in \mathcal{T}} U_t$ the total number of available bins of any type.

The item-to-bin accommodation rules of the GBPP are 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 bin volume;
- The number of bins used for each type $t \in \mathcal{T}$ must be within the lower and upper availability limits L_t and U_t;
- The total number of used bins cannot exceed the total number of available bins U.

Infeasibility may arise when the available bins are not sufficient to load all compulsory items. To address this issue, we add a special bin v of volume $W_v = \sum_{i \in \mathcal{I}^{\mathrm{C}}} w_i$, thus able to load all compulsory items, and set its cost C_v to a value much higher than the costs of the remaining bins in order to discourage its use, e.g., $C_v \gg \sum_{t \in \mathcal{T}} C_t$.

3.2. Assignment formulation of the GBPP

Consider the following decision variables:

- Bin selection binary variables y_j equal to 1 if bin $j \in \mathcal{J}$ is used, 0 otherwise;
- 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.

An assignment model of the GBPP can then be formulated as follows:

$$\text{Minimize} \quad \sum_{j\in\mathcal{J}} C_j y_j - \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}^{\text{NC}}} p_i x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{i\in\mathcal{I}} w_i x_{ij} \leq W_j y_j \qquad j \in \mathcal{J} \tag{2}$$

$$\sum_{j\in\mathcal{J}} x_{ij} = 1 \qquad i \in \mathcal{I}^{\text{C}} \tag{3}$$

$$\sum_{j\in\mathcal{J}} x_{ij} \leq 1 \qquad i \in \mathcal{I}^{\text{NC}} \tag{4}$$

$$\sum_{j\in\mathcal{J}:\sigma(j)=t} y_j \leq U_t \qquad t \in \mathcal{T} \tag{5}$$

$$\sum_{j\in J:\sigma(j)=t} y_j \geq L_t \qquad t \in \mathcal{T} \tag{6}$$

$$\sum_{j\in\mathcal{J}} y_j \leq U \tag{7}$$

$$y_j \in \{0,1\}, \qquad j \in \mathcal{J} \tag{8}$$

$$x_{ij} \in \{0,1\}, \qquad i \in \mathcal{I}, \quad j \in \mathcal{J} \tag{9}$$

The objective function (1) minimizes the total net cost of the packing, given by the difference between the total cost of the used bins and the total profit of the selected non-compulsory items. The profit of the compulsory items is not considered in the objective function because it is a constant. Regarding the type of optimization, we choose to present the minimization version to follow the tradition of packing problems. The equivalent formulation obtained by maximizing the total net profit (total profit minus total cost) would recall the knapsack problem setting.

Constraints (2) have the double effect of linking the usage of bins to the accommodation of items and bounding the capacity of each used bin. Constraints (3) and (4) ensure that each compulsory and not-compulsory item is loaded into exactly one and at most one bin, respectively. Constraints (5) and (6) enforce the maximum and minimum number of available bins of each type, while Constraints (7) limit the total number of selected bins. Constraints (8) and (9) enforce the integrality nature of the decision variables. Notice that, Constraints (5) and (6) could be implicitly managed, the former by limiting the number of y_j variables to U_t for type t (i.e., defining the appropriate number of bins only), and the latter by setting $y_j = 1$ for $j = 1, \ldots, L_t$. We prefer to keep the constraints in the formulation, however, for consistency with the set covering model of Section 3.4.

The assignment model (1)-(9) is named *AM* and its continuous relaxation *R − AM*. *AM* involves a polynomial number of variables and constraints. It is not suitable for developing efficient algorithms, however, due to the significant solution-space symmetry of the item-to-bin assignment variables, which is typical of these packing models. Yet, as mentioned above, *AM* is the starting point to compute our first lower bound named LB_1. Furthermore, *AM* is also suitable to show how the GBPP generalizes some classic packing and knapsack problems. This issue is addressed next.

3.3. Generalization of classic packing and knapsack problems

The assignment formulation of the GBPP (1)-(9) generalizes a number of classic packing problems, including the BPP [16], which may be obtained by considering a single bin type with $C_j = 1$, $j \in \mathcal{J}$ and $\mathcal{I}^{\text{NC}} = \emptyset$, i.e., all items must be loaded. Constraints (4) - (6) are then redundant and the objective function becomes the minimization of the number of bins, which is characteristic of the BPP.

Allowing several bin types and $\mathcal{I}^{\text{NC}} = \emptyset$ yields the VCSBPP, where the total cost of the selected bins $\sum_{j\in\mathcal{J}} C_j y_j$ is minimized (Constraint (4) becomes redundant) [17, 5]. Notice that, an equivalent formulation for the VCSBPP can be obtained by imposing $\mathcal{I}^{\text{C}} = \emptyset$ and setting the item profit higher than the cost of any bin type, $p_i > \max_{j\in\mathcal{J}} C_j$, which makes any bin profitable even when only one item is loaded into it.

The GBPP can similarly generalize a number of knapsack problems. Specifically, the GBPP reduces to the KP by setting $|\mathcal{T}| = 1$, $|\mathcal{J}| = 1$, and $\mathcal{I}^{\text{C}} = \emptyset$. The MKP can be modeled by setting $|\mathcal{T}| > 1$, $|\mathcal{J}| = m$, where m is the number of knapsacks, and $\mathcal{I}^{\text{C}} = \emptyset$. Finally, the MKPI is addressed by setting $|\mathcal{T}| = 1$, $|\mathcal{J}| = m$, and $\mathcal{I}^{\text{C}} = \emptyset$.

3.4. Set Covering formulation of the GBPP

We introduce a set covering formulation of the GBPP based on feasible loading patterns for the bins.

A *feasible loading pattern* k_t for a bin of type $t \in \mathcal{T}$ is a set of items that may be loaded into the bin while satisfying all dimension restrictions and accommodation rules. Let $\mathcal{K}_t = \{k_t\}$ be the set of all feasible loading patterns for bin type $t \in \mathcal{T}$. A feasible loading pattern k_t is described by a vector $\boldsymbol{a_{k_t}}$ of indicator functions $a^i_{k_t},\ i \in \mathcal{I},\ k_t \in \mathcal{K}_t,\ t \in \mathcal{T}$, such that $a^i_{k_t} = 1$ if item i is packed into pattern k_t, 0 otherwise. The cost of pattern k_t is then the difference between the cost of the bin type t and the total profit of non-compulsory items in that pattern:

$$c_{k_t} = C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i a^i_{k_t}. \tag{10}$$

We define the bin loading pattern selection variables λ_{k_t} equal to 1 if pattern $k_t \in \mathcal{K}_t$ is used, 0 otherwise. The set covering formulation of the GBPP may then be written as follows:

$$\begin{aligned}
\text{Minimize} \quad & \textstyle\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} c_{k_t}\lambda_{k_t} && && (11)\\
\text{Subject to} \quad & \textstyle\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} a^i_{k_t}\lambda_{k_t} = 1 \quad i \in \mathcal{I}^{\mathrm{C}} && \text{(dual variable } \mu_i \text{ free)} && (12)\\
& \textstyle\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} a^i_{k_t}\lambda_{k_t} \le 1 \quad i \in \mathcal{I}^{\mathrm{NC}} && \text{(dual variable } \nu_i \le 0) && (13)\\
& \textstyle\sum_{k\in\mathcal{K}_t} \lambda_{k_t} \le U_t \quad t \in \mathcal{T} && \text{(dual variable } \alpha_t \le 0) && (14)\\
& \textstyle\sum_{k\in\mathcal{K}_t} \lambda_{k_t} \ge L_t \quad t \in \mathcal{T} && \text{(dual variable } \beta_t \ge 0) && (15)\\
& \textstyle\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} \lambda_{k_t} \le U && \text{(dual variable } \epsilon \le 0) && (16)\\
& \lambda_{k_t} \in \{0,1\}, \quad k_t \in \mathcal{K}_t,\ t \in \mathcal{T} && && (17)
\end{aligned}$$

The objective function (11) minimizes the total cost of the selected bin loading patterns. Since the feasibility of the item-to-bin accommodation is guaranteed by the feasibility of the loading patterns, the constraints equivalent to (2) in the model *AM* are not required anymore. Constraints (12)-(16) have the same meaning as (3)-(7), and (17) are the integrality constraints.

The set covering model (11)-(17) is named *SC* and its continuous relaxation $R-SC$. *SC*, when compared to *AM*, has the advantage to separate the feasibility phase from the optimality one. The feasibility phase is already addressed by the pattern generation, whilst the model is only devoted to find an optimal combination of patterns.

Whilst of course models *AM* and *SC* have the same optimum, the optima of $R-AM$ and $R-SC$ are generally different. In the following, we prove that the lower bound to the GBPP obtained by optimizing $R-SC$ is not weaker than that obtained by optimizing $R-AM$.

Property 1. *Given any solution x_2 of $R-SC$ (which is feasible by construction), a corresponding feasible solution $x_1(x_2)$ of $R-AM$ can be built as follows. For any $\lambda_{k_t} = 1$ of x_2 assign in $R-AM$ the items of pattern k_t to bin j_{k_t} by putting in the solution $x_1(x_2)$ $x_{ij_{k_t}} = a^i_{k_t}\lambda_{k_t}$. The two solutions have the same value.*

Proof. Trivial. □

Theorem 1. *Let $L_{R-AM} = optimum(R-AM)$ and $L_{R-SC} = optimum(R-SC)$, then $L_{R-AM} \le L_{R-SC}$.*

Proof. By contradiction, let us suppose that an instance of $R-AM$ such that $L_{R-AM} > L_{R-SC}$ there exists. Let $\overline{x}_2$ be the optimal solution associated to L_{R-SC}. By using Property 1, we can build a solution $\overline{x}_1(\overline{x}_2)$ of $R-AM$ with value L_{R-SC}, which contradicts the optimality of L_{R-AM}.

□

4. Lower bounds

We introduce two lower bounds that can be computed starting from each of the two problem formulations. The assignment model *AM* is the basis for a lower bound which can be calculated by solving an Aggregate Knapsack Problem (AKP). We show how to derive the AKP from the model *AM* in Section 4.1.

The second lower bound is derived from the set covering formulation *SC* and is calculated by applying a column generation technique where, at each step, a new feasible pattern (i.e., a new column for the restricted master problem) is found by solving a knapsack problem (see Section 4.2).

4.1. Lower bound through the Aggregate Knapsack Problem

To derive an Aggregate Knapsack Problem from the assignment model *AM* presented in Section 3.2, we aggregate Constraints (2) into a unique inequality by summing them up. We thus consider an aggregate knapsack, which may be thought of as a unique large bin with volume equal to the total volume of the bins. We have

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} w_i x_{ij} \leq \sum_{j \in \mathcal{J}} W_j y_j \quad \Longrightarrow \quad \sum_{i \in \mathcal{I}^C} w_i \sum_{j \in \mathcal{J}} x_{ij} + \sum_{i \in \mathcal{I}^{NC}} w_i \sum_{j \in \mathcal{J}} x_{ij} \leq \sum_{j \in \mathcal{J}} W_j y_j.$$

Note that, by (3), for any compulsory item i, $\sum_{j \in \mathcal{J}} x_{ij} = 1$, whilst, for any non-compulsory item i, the variable x_{ij} can be reduced to x_i, which states whether item i is put into the aggregate knapsack or not. Therefore, we have

$$\sum_{i \in \mathcal{I}^C} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j. \tag{18}$$

We drop Constraints (5) and (6) by managing them implicitly as indicated in Section 3.2. Constraints (7) are kept, as one cannot implicitly manage them. A first lower bound, named LB_1, can then be found by solving the following AKP

$$\text{Minimize} \quad \sum_{j \in \mathcal{J}} C_j y_j - \sum_{i \in \mathcal{I}^{NC}} p_i x_i \tag{19}$$

$$\text{Subject to} \quad \sum_{i \in \mathcal{I}^C} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j \tag{20}$$

$$\sum_{j \in \mathcal{J}} y_j \leq U \tag{21}$$

$$y_j \in \{0, 1\}, \qquad j \in \mathcal{J} \tag{22}$$

$$x_i \in \{0, 1\}, \qquad i \in \mathcal{I} \tag{23}$$

Since LB_1 comes from the resolution of the AKP, we also refer to it as an *aggregate knapsack lower bound.* Note that, when all items are compulsory, one can use bounds from the literature. In this case, (19)-(23) reduces to

$$\text{Minimize} \quad \sum_{j \in \mathcal{J}} C_j y_j \tag{24}$$

$$\text{Subject to} \quad \sum_{i \in \mathcal{I}^C} w_i \leq \sum_{j \in \mathcal{J}} W_j y_j \tag{25}$$

$$\sum_{j \in \mathcal{J}} y_j \leq U \tag{26}$$

$$y_j \in \{0, 1\}, \qquad j \in \mathcal{J} \tag{27}$$

which is the model used by Crainic et al. [5] to compute lower bounds to the VCSBPP.

4.2. Lower bound through column generation

This lower bound is computed from the $R-SC$ model through column generation and is named LB_2. It is widely used in packing problems [1, 22] and provides the means to implicitly deal with a large number of variables.

The column generation approach applied to the $R-SC$ model consists in starting with a relatively small set of feasible patterns $\mathcal{P}$, which correspond to a restricted problem named $R-SC_R$. First we solve $R-SC_R$ and then attempt to generate new feasible patterns with negative reduced cost. If successful, these are added to $\mathcal{P}$ and the procedure is restarted. Otherwise, we have found an optimal solution of $R-SC$ and the procedure stops with a lower bound to the GBPP.

The main steps of the procedure are as follows:

1. Find an initial feasible solution of the GBPP and the corresponding set $\mathcal{P}$;
2. Solve to optimality $R-SC_R$ and let L_{R-SC_R} be its optimum;
3. For each bin type $t \in \mathcal{T}$
   (a) Find the pattern variable $\lambda_{\bar{k}_t}$, $\bar{k}_t \in \mathcal{K}_t$ with the smallest reduced cost $r_{\bar{k}_t}$, among all non-basic pattern variables λ_{k_t} of the optimal solution of $R-SC_R$;
   (b) If $r_{\bar{k}_t} < 0$, $\mathcal{P} = \mathcal{P} \cup \{\lambda_{\bar{k}_t}\}$;
4. If $r_{\bar{k}_t} \geq 0$ for all bin types t, then stop and L_{R-SC_R} is the lower bound to the GBPP, otherwise, go to 2.

Note that, in Step 3, the procedure adds at most $|\mathcal{T}|$ columns to $\mathcal{P}$ at each iteration.

The main issue is how to find negative reduced cost feasible patterns. Consider the dual variables associated to the constraints of the $R-SC_R$ model (see (11)-(17)). The reduced cost r_{k_t} of a given pattern variable λ_{k_t} is $c_{k_t} - \left[\boldsymbol{\mu}^T \ \boldsymbol{\nu}^T\right] \boldsymbol{a}_{\boldsymbol{k_t}} - \left[\boldsymbol{\alpha}^T \ \boldsymbol{\beta}^T \ \epsilon\right] \overline{\mathbf{1}}_{\boldsymbol{t}}$, where $\boldsymbol{a}_{\boldsymbol{k_t}} = [a^i_{k_t}]$ and $\overline{\mathbf{1}}_{\boldsymbol{t}}$ is a vector of size $2|\mathcal{T}|+1$, with 1 in the rows corresponding to bin type t and in the last row, 0 otherwise. By (10), we expand this expression as follows

$$\begin{aligned} r_{k_t} &= C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i \, a^i_{k_t} - \left[\boldsymbol{\mu}^T \ \boldsymbol{\nu}^T\right] \boldsymbol{a}_{\boldsymbol{k_t}} - \left[\boldsymbol{\alpha}^T \ \boldsymbol{\beta}^T \ \epsilon\right] \overline{\mathbf{1}}_{\boldsymbol{t}} \\ &= C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i \, a^i_{k_t} - \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a^i_{k_t} - \alpha_t - \beta_t - \epsilon \\ &= C_t - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} (p_i + \nu_i) \, a^i_{k_t} - \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} - \alpha_t - \beta_t - \epsilon. \end{aligned} \tag{28}$$

We now define a column generation subproblem. Given a bin of type $t \in \mathcal{T}$, this subproblem finds the non-basic pattern with the minimum reduced cost. Note that, the vector $\boldsymbol{a}_{\boldsymbol{k_t}}$ defining a not-yet-generated pattern $k_t \in \mathcal{K}_t$ is not known, but it may be expressed in terms of the item-to-bin assignment variable x_i, which is equal to 1 if item $i \in \mathcal{I}$ belongs to the pattern, 0 otherwise. Since the dual variables α_t, β_t, and ϵ, as well as the bin cost C_t, are constant for any given bin type $t \in \mathcal{T}$, then finding the feasible pattern with the minimum reduced cost for bin type $t \in \mathcal{T}$ becomes the following knapsack problem

$$\text{Maximize} \qquad \left\{ \sum_{i \in \mathcal{I}^{\mathrm{NC}}} (p_i + \nu_i) \, x_i + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, x_i \right\} \tag{29}$$

$$\text{Subject to:} \qquad \sum_{i \in \mathcal{I}} w_i x_i \leq W_t \qquad t \in \mathcal{T} \tag{30}$$

$$x_i \in \{0, \ 1\} \qquad i \in \mathcal{I} \tag{31}$$

Any feasible solution may be used to initialize the procedure, including the trivial solution obtained by loading each compulsory item into a different bin. More accurate heuristics are presented in Section 5.

Finally, note that a better lower bound can be obtained by taking the maximum between the two previous lower bounds. We call this new lower bound $LB_3 = \max\{LB_1, \ LB_2\}$.

5. Upper bounds

We derive several upper bounds to the GBPP 1) through constructive heuristics; 2) by making feasible some of the GBPP lower bounds; 3) through column generation-based heuristics.

5.1. Upper bounds through constructive heuristics

We propose constructive heuristics to load items into bins based either on the *First Fit Decreasing* (FFD) or the *Best Fit Decreasing* (BFD) heuristics for the BPP, with different sorting rules for items and bins. We briefly recall how FFD and BFD work. Starting with items sorted by non-increasing volume, FFD loads items one after the other into the first bin where they fit. BFD attempts to load each item into the "best" bin, usually the bin with the minimum *free volume* after loading the item. The free volume is defined as the bin volume minus the total volume of the loaded items. Both heuristics create a new bin when an item cannot be accommodated into the existing ones. Despite their simplicity, the FFD and BFD heuristics offer good performances for the BPP [16].

Note that, whilst in the BPP items are sorted by non-increasing volume, many item and bin sorting rules are possible for the GBPP, due to the presence of multiple attributes. We exploit this characteristic in building our heuristics.

Given the sorted lists of items and bins, *SIL* and *SBL* (see Section 5.1.1 for deriving these lists), our heuristics are composed of three main components displayed in Algorithms 1, 2, and 3. Given a list S of selected bins (initially empty), for each item belonging to *SIL*, Algorithm 1 (the MAIN procedure) looks for the first or the best bin in S able to load such an item by using FFD or BFD, respectively. If this bin exists, the item is loaded into it, otherwise a new bin is possibly selected. Actually, a new bin is selected when the item is compulsory otherwise, we evaluate whether to select a new bin or not. This evaluation is performed by Algorithm 2 (the PROFITABLE procedure), which measures the profitability of the current item. In particular, a new bin will be selected and the item will be loaded into it if the profit of this item plus the profits of the remaining non-compulsory items in SIL is greater than the cost of the new bin. Finally, the POST-OPTIMIZATION procedure of Algorithm 3 attempts to improve the final solution by evaluating possible bin swaps that replace loaded bins with available cheaper bins of sufficient capacity.

Algorithm 1 The MAIN procedure

$\mathcal{S} := \emptyset$
for all $i \in SIL$ **do**
  Identify the bin $b \in \mathcal{S}$ into which item i can be loaded:
  - FF: the first bin with enough empty volume to accommodate item i
  - BF: the bin with the minimum free volume after loading item i

  if b exists **then**
    Load item i into bin b
    $\mathcal{S} := \mathcal{S} \cup \{b\}$
  else
    if $i \in I^C$ **then**
      Identify the first bin $b \in SBL \setminus \mathcal{S}$ such that $w_i \leq W_b$.
      Load item i into bin b
      $\mathcal{S} := \mathcal{S} \cup \{b\}$
    else
      Identify the bin $b \in SBL \setminus \mathcal{S}$ such that PROFITABLE(i, b) returns TRUE
      if b exists **then**
        Load item i into bin b
        $\mathcal{S} := \mathcal{S} \cup \{b\}$
      else
        reject item i
      end if
    end if
  end if
end for
POST-OPTIMIZATION

Algorithm 2 The PROFITABLE procedure for new bin selection

SIL_i : sublist of SIL starting from the item i;
Load i into b and initialize the bin profit $P_b = p_i$;
for all $i' \in SIL_i$ **do**
  if i' can be loaded into b **then**
    Load i' into b and update the bin profit $P_b = P_b + p_{i'}$;
  end if
  if $P_b > c_b$, return TRUE **else** return FALSE.
end for

Algorithm 3 The POST-OPTIMIZATION procedure

for all $j \in \mathcal{S}$ **do**
  for all $k \in \mathcal{J} \setminus \mathcal{S}$ **do**
    $U_j = \sum_{i \text{ loaded into } j} w_i$
    if $W_k \geq U_j$ and $C_k < C_j$ **then**
      Move all the items from j to k
      $\mathcal{S} = \mathcal{S} \setminus \{j\} \cup \{k\}$
    end if
  end for
end for

5.1.1. Building the sorted lists of items and bins

We define four sorting rules to embed, and test, into the constructive heuristics we propose. All the rules have in common that compulsory items are sorted at the top of the item list by non-increasing volume. The four rules are:

1. ***Bins:*** Non-decreasing C_j/W_j and non-decreasing volumes W_j;
   Non-compulsory items: Non-increasing p_i/w_i and non-increasing volumes w_i;
2. ***Bins:*** Non-decreasing C_j/W_j and non-decreasing volumes W_j;
   Non-compulsory items: Non-increasing volumes w_i and non-increasing p_i/w_i;
3. ***Bins:*** Non-decreasing C_j/W_j and non-increasing volumes W_j;
   Non-compulsory items: Non-increasing p_i/w_i and non-increasing volumes w_i;
4. ***Bins:*** Non-decreasing C_j/W_j and non-increasing volumes W_j;
   Non-compulsory items: Non-increasing volumes w_i and non-increasing p_i/w_i.

5.2. Upper bounds through the lower bound LB_1

To derive an upper bound from LB_1, introduced in Section 4.1, we apply our constructive heuristics to the two ordered lists of bins and items. We name this approach *lower bound-based constructive heuristics*. We refer to the FFD- (L-FFD) or BFD-based lower bound (L-BFD) when the constructive heuristics implements the FFD or the BFD principle, respectively.

The ordered lists of items and bins are obtained as follows. An optimal solution of the AKP model (20)-(23) consists in a set of non-compulsory items, $\mathcal{I}_{agg}$, and a set of bins, $\mathcal{J}_{agg}$. $\mathcal{I}_{agg}$ contains the non-compulsory items associated to the variables x_i equal to 1 in the optimal solution of the AKP. Similarly, $\mathcal{J}_{agg}$ contains the bins associated to the variables y_j equal to 1 in the optimal solution of the AKP. We then extract two subsets $\mathcal{I}'_{agg}$ and $\mathcal{J}'_{agg}$ by selecting δ^i % of items in $\mathcal{I}_{agg}$ and δ^b% of bins in $\mathcal{J}_{agg}$, respectively. We then randomly select a particular sorting rule s among the four ones, and order the item and bin lists as follows:

- Order the items in $\mathcal{I}'_{agg}$ according to s and put them at the top of the item list, before any non-compulsory item. Then, order the remaining item $\mathcal{I} \setminus \mathcal{I}'_{agg}$ according to s;
- Order the bins in $\mathcal{J}'_{agg}$ according to s and put them at the top of the bin list. Then, order the remaining bins $\mathcal{J} \setminus \mathcal{J}'_{agg}$ according to s.

To state that L-FFD and L-BFD are characterized by δ^i, δ^b, and s, we write L-FFD$\left(\delta^i, \delta^b, s\right)$ and L-BFD$\left(\delta^i, \delta^b, s\right)$. The values of δ^i and δ^b are obtained by calibration (see Section 6.3 for details), whilst $s \in \{1, 2, 3, 4\}$, according to

the four sorting strategies presented in Section 5.1.1.

5.3. Upper bounds through column generation-based heuristics

We present two approaches to compute upper bounds starting from the column generation-based solution of the relaxation $R - SC$ of the set covering model SC (11)-(17).

The first approach is to solve SC exactly, e.g., by branch-&-bound, considering only the columns obtained by the column-generation procedure while computing the lower bound. This may still be quite time consuming, however. Consequently, we might stop with the branch-&-bound after a given computing time and name Z_{SC} the resulting value of the objective function, which is an upper bound to the GBPP.

The second approach is based on *diving*, a well-known method for finding good quality integer solutions from the optimal continuous solutions [2]. The working principle is to iteratively round up variables and re-optimize the continuous relaxation.

The diving heuristics assumes that the optimal bin loading patterns of the GBPP are in the restricted set corresponding to the $R - SC_R$, and iteratively slightly perturbs the optimal continuous solution by fixing to integer some pattern-selection variables. The key feature is how to choose the variables to be fixed. Two strategies are obtained by selecting among the non-integral pattern variables the ones which maximize the expressions (32) and (33) This generates two diving heuristics named Diving1 and Diving2, which are included in the final comparative experiments of Section 6:

$$\sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a^i_{k_t} + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} \tag{32}$$

$$(1 - \lambda_{k_t}) \left(\sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a^i_{k_t} + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a^i_{k_t} \right) \tag{33}$$

6. Computational results

The goal of the numerical experiments is to explore the performance of the proposed lower and upper bound procedures. Section 6.1 introduces the instance sets, whilst detailed results of different variants of the lower and upper bound procedures are given in Sections 6.2 and 6.3. We study the impact of a number of problem parameters on our best bounds in Section 6.4.

6.1. Instance classes

No instances are present in the literature for the GBPP. We generated instances, partially based on those for the VSBPP and the BPP [17, 22, 4, 5]. The instances are grouped into 5 classes:

- Class 0: 300 instances by Monaci [17]. We chose Monaci's instances because they are more challenging than Correia's [4], as shown in [5]. Since these instances were conceived for the VSBPP, all items of each instance are compulsory. Ten instances were randomly generated for each combination of number of items, item profit, item volume, and bin type for a total of 300 instances. For the sake of completeness, we report here the details of Monaci's instances:
  - Number of items: 25, 50, 100, 200, and 500;
  - Item volume: **I1:** [1, 100]; **I2:**[20, 100]; **I13:**[50, 100];
  - Item profit: not defined because all items are compulsory;
  - Bin type:
    - 3 types of bins, with volumes 100, 120, and 150, respectively, and costs equal to volumes;
    - 5 types of bins, with volumes 60, 80, 100, 120, and 150, respectively, and costs equal to volumes.

  For each bin type t, $L_t = 0$ and $U_t = \lceil V_{tot}/V_t \rceil$, where V_{tot} is the total item volume. No values for the total number of available bins U are given.
- Class 1: same instances of Class 0, but with all items non-compulsory and item profits generated according to the $p_i \in \lceil \mathcal{U}(0.5, 3) w_i \rceil$ uniform distribution.
- Class 2: same instances of Class 0, but with all items non-compulsory and item profit generated according to the $p_i \in \lceil \mathcal{U}(0.5, 4) w_i \rceil$ uniform distribution.

- Class 3: a selection of 12 large instances (500 items) from Class 1 and Class 2 with a representative mix of characteristics in terms of item volume, item profit, and bin types. For each instance, we randomly derived five more instances with 0%, 25%, 50%, 75%, and 100% of compulsory items, for a total of 60 instances.
- Class 4: the aim of this class is to study the behavior of Constraints (7) and (16) on the total number of available bins U. Thus, we select 24 instances from Class 1 and Class 2. For each instance, we first computed the number of bins $\overline{U}$ employed by the BFD constructive heuristics. We then solved the GBPP varying the value of U as a percentage of $\overline{U}$

$$U = \overline{U}(1 - X), \quad X \in \{0, 0.1, 0.2, 0.3\}, \tag{34}$$

  All these combinations make up Class 4 with 96 instances.

The algorithms were coded in C++ and the models implemented with CPLEX 12.1 [11]. The upper bound Z_{SC} was computed using Gurobi 4.0, due to its efficiency in finding good feasible solutions within a quite limited computing time (20 seconds) [10]. Experiments were made on a Pentium IV 3.0 GHz workstation with 4 GB of RAM.

6.2. Lower bounds

Table 1 shows the lower bound results, comparing the performance of the three proposed lower bounds, LB_1, LB_2, and LB_3), relative to Z_{SC}, the best upper bound (Section 5.3) or to the known optimum solution [Class 0 instances, named in the following Monaci optima 17]. Columns 1 to 3 display the instance class, number of bin types, and number of items, respectively. Columns 4 and 5, 6 and 7, and 8 and 9 display the mean percentage gap to Z_{SC} or the known optimum, and the number of optima achieved by LB_1, LB_2, and LB_3, respectively. Each row of Table 1 gives the results of 30 instances (3 item volume types, I1, I2, and I3, times 10 repetitions). For each class and globally, the table also displays the respective average gaps and the total number of optima attained (and the corresponding percentage with respect to the total number of instances).

Table 1 reports very promising results. The overall percentage gap for LB_3 is quite tight (0.08%) and almost half (46%) of the instances are solved to optimality.

6.3. Upper bounds

Table 2 displays comparative results for the constructive heuristics upper bounds. The first three columns display the same type of information as previously. Columns 4 to 7 and 8 to 11 display relative-gap results with respect to LB_3 (except for Class 0 instances with Monaci optima) for the FFD and the BFD procedures, respectively, using the four item and bin sorting rules of Section 5.1.1.

The results summed up in Table 2 show that BFD offers slightly better results than FFD. Furthermore, we see that BFD(3) is the best performing constructive heuristics.

We compare the remaining upper bounds, i.e., those obtained through the lower bound LB_1 (Section 5.2) and those derived from the column generation-based heuristics (Section 5.3) in Table 3. Column 1 shows the instance class, Column 2 the number of bin types, and Column 3 the number of items. For the remaining columns of Table 3, we have:

BFD(3): BFD heuristics with the third sorting rule;

L-BFD(1, 0.1, 3): Upper bound obtained from the lower bound LB1 with the constructive heuristics BFD(3) and $\delta^i = 1$, $\delta^b = 0.1$. The values of δ^i and δ^b were obtained by calibration performed by running L-BFD$\left(\delta^i, \delta^b, 3\right)$ on a number of selected instances, and varying δ^i and δ^b from 0.1 to 1 with a step of 0.1. The pair $\left(\delta^i, \delta^b\right)$ which gave the minimum mean gap was then selected;

C-BFD(3)= $\min\left\{\text{BFD(3)}, \min_{\delta^i \in \Delta^i, \delta^b \in \Delta^b}\left\{\text{L-BFD}\left(\delta^i, \delta^b, 3\right)\right\}\right\}$, where $\Delta^i = \Delta^b = \{0.1, 0.2, 0.3\}$; These value combinations provided low mean gaps during the calibration phase;

Z_{SC} : Upper bound obtained through the column generation-based heuristics;

DIVE(1), DIVE(2): Upper bounds obtained through the column generation-based heuristics using the diving strategies Diving1 and Diving2, respectively (Section 5.3);

B-DIVE(1)= Minimum {BFD(3), DIVE(1) };

B-DIVE(2)= Minimum {BFD(3), DIVE(2) }.

			LB_1		LB_2		LB_3	
CLASS	**BINS**	**ITEMS**	**% GAP**	**OPT**	**% GAP**	**OPT**	**% GAP**	**OPT**
0	3	25	1.21	10	0.31	13	0.18	21
		50	0.65	12	0.21	5	0.13	16
		100	0.51	16	0.07	8	0.04	22
		200	0.31	19	0.04	5	0.02	23
		500	0.31	20	0.03	3	0.01	23
	5	25	0.80	10	0.20	13	0.12	20
		50	0.51	15	0.12	9	0.05	21
		100	0.49	18	0.07	6	0.02	22
		200	0.27	20	0.04	6	0.01	24
		500	0.25	20	0.01	6	0.00	24
			0.53	**160 (53%)**	**0.11**	**74 (25%)**	**0.06**	**216 (72%)**
1	3	25	2.16	4	0.27	16	0.19	20
		50	0.86	1	0.17	6	0.15	5
		100	0.72	2	0.12	3	0.11	5
		200	0.45	1	0.13	6	0.12	7
		500	0.33	0	0.10	3	0.10	3
	5	25	1.42	5	0.18	13	0.13	16
		50	0.75	3	0.10	12	0.09	13
		100	0.57	5	0.04	10	0.03	13
		200	0.29	4	0.03	6	0.02	9
		500	0.29	2	0.08	6	0.07	8
			0.78	**27 (9%)**	**0.12**	**81 (27%)**	**0.10**	**99 (33%)**
2	3	25	1.31	5	0.18	14	0.16	17
		50	0.61	4	0.12	4	0.10	8
		100	0.41	3	0.06	7	0.06	8
		200	0.30	1	0.10	5	0.10	5
		500	0.26	0	0.08	4	0.07	4
	5	25	0.98	4	0.12	14	0.09	16
		50	0.52	8	0.06	8	0.05	15
		100	0.40	6	0.04	6	0.03	11
		200	0.18	4	0.05	5	0.04	9
		500	0.19	1	0.05	5	0.05	6
			0.52	**36 (12%)**	**0.09**	**72 (24%)**	**0.07**	**99 (33%)**
OVERALL			**0.61**	**223 (25%)**	**0.11**	**227 (25%)**	**0.08**	**414 (46%)**

Table 1: Lower bound results

CLASS	BINS	ITEMS	FFD(1) % GAP	FFD(2) % GAP	FFD(3) % GAP	FFD(4) % GAP	BFD(1) % GAP	BFD(2) % GAP	BFD(3) % GAP	BFD(4) % GAP
0	3	25	12.80	12.80	3.68	3.68	12.80	12.80	3.47	3.47
		50	13.25	13.25	2.44	2.44	13.25	13.25	2.36	2.36
		100	12.21	12.21	1.66	1.66	12.21	12.21	1.62	1.62
		200	10.28	10.28	1.28	1.28	10.28	10.28	1.25	1.25
		500	8.97	8.97	1.08	1.08	8.97	8.97	1.07	1.07
	5	25	10.49	10.49	1.93	1.93	10.49	10.49	1.93	1.93
		50	11.59	11.59	1.79	1.79	11.59	11.59	1.78	1.78
		100	10.63	10.63	1.27	1.27	10.63	10.63	1.26	1.26
		200	10.82	10.82	0.83	0.83	10.82	10.82	0.82	0.82
		500	10.44	10.44	0.65	0.65	10.44	10.44	0.63	0.63
			11.15	**11.15**	**1.66**	**1.66**	**11.15**	**11.15**	**1.62**	**1.62**
1	3	25	13.18	17.03	4.07	8.89	12.89	16.67	3.96	8.72
		50	13.62	17.93	3.22	7.59	13.57	17.87	3.20	7.54
		100	12.67	16.60	2.18	7.36	12.55	16.52	2.16	7.34
		200	11.30	15.08	1.47	7.15	11.26	15.05	1.45	7.13
		500	9.83	13.44	0.94	6.58	9.81	13.46	0.94	6.58
	5	25	9.74	14.86	3.38	8.40	9.79	14.54	3.33	8.37
		50	10.56	16.12	3.49	7.60	10.56	15.88	3.46	7.56
		100	9.53	14.41	1.99	6.41	9.49	14.32	1.97	6.39
		200	9.57	14.84	1.48	6.28	9.55	14.77	1.49	6.27
		500	9.36	14.68	1.00	6.32	9.36	14.65	1.00	6.32
			10.93	**15.50**	**2.32**	**7.26**	**10.88**	**15.38**	**2.30**	**7.22**
2	3	25	9.50	10.25	3.45	4.63	9.48	10.22	3.17	4.49
		50	10.91	11.99	2.39	4.58	10.85	11.93	2.32	4.54
		100	9.42	10.83	1.43	3.54	9.35	10.79	1.41	3.53
		200	8.35	9.57	1.20	3.37	8.31	9.58	1.20	3.33
		500	7.37	8.81	0.77	3.30	7.33	8.82	0.77	3.29
	5	25	7.18	9.36	3.28	3.86	7.18	9.26	3.35	3.86
		50	7.45	9.65	2.47	3.33	7.47	9.57	2.31	3.32
		100	6.77	9.11	1.73	3.59	6.75	9.08	1.70	3.58
		200	6.78	9.30	1.13	3.26	6.77	9.25	1.12	3.25
		500	6.56	9.16	0.77	3.13	6.56	9.16	0.77	3.13
			8.03	**9.80**	**1.86**	**3.66**	**8.00**	**9.77**	**1.81**	**3.63**
OVERALL			**10.04**	**12.15**	**1.95**	**4.19**	**10.01**	**12.10**	**1.91**	**4.16**

Table 2: Constructive heuristics upper bounds

CLASS	BINS	ITEMS	BFD(3) % GAP	L-BFD(1, 0.1, 3) % GAP	C-BFD(3) % GAP	Z_{SC} % GAP	DIVE(1) % GAP	DIVE(2) % GAP	B-DIVE(1) % GAP	B-DIVE(2) % GAP
0	3	25	3.47	3.35	1.94	0.26	1.95	2.02	1.31	1.54
		50	2.36	2.55	1.84	0.19	1.29	1.39	0.95	1.05
		100	1.62	1.53	1.15	0.16	1.85	1.45	0.84	0.66
		200	1.25	1.34	1.02	0.18	1.75	1.15	0.59	0.51
		500	1.07	0.85	0.68	0.31	2.12	1.04	0.59	0.51
	5	25	1.93	2.49	1.75	0.13	1.27	0.88	0.68	0.60
		50	1.78	2.56	1.68	0.06	0.79	0.55	0.63	0.52
		100	1.26	1.84	1.17	0.03	0.59	0.45	0.42	0.39
		200	0.82	1.57	0.82	0.06	0.64	0.45	0.34	0.28
		500	0.63	1.25	0.63	0.05	0.76	0.41	0.16	0.14
			1.62	**1.93**	**1.27**	**0.14**	**1.30**	**0.98**	**0.65**	**0.62**
1	3	25	3.96	3.85	2.95	0.19	1.30	1.42	0.87	0.89
		50	3.20	2.98	2.46	0.15	1.05	0.94	0.88	0.82
		100	2.16	2.37	1.98	0.11	0.81	2.18	0.66	0.68
		200	1.45	1.77	1.37	0.12	1.57	2.28	0.55	0.61
		500	0.94	1.14	0.92	0.10	1.89	2.79	0.40	0.55
	5	25	3.33	3.67	2.42	0.12	0.74	0.76	0.55	0.58
		50	3.46	3.85	2.92	0.09	0.37	0.52	0.37	0.52
		100	1.97	2.47	1.83	0.03	0.53	0.52	0.35	0.36
		200	1.49	1.92	1.45	0.02	0.55	1.11	0.28	0.32
		500	1.00	1.21	0.97	0.07	0.73	1.18	0.15	0.18
			2.30	**2.52**	**1.93**	**0.10**	**0.95**	**1.37**	**0.51**	**0.55**
2	3	25	3.17	3.10	2.09	0.16	1.99	2.07	1.05	0.95
		50	2.32	2.70	2.09	0.10	0.83	1.25	0.77	0.84
		100	1.41	1.67	1.32	0.06	0.98	1.64	0.38	0.48
		200	1.20	1.49	1.15	0.10	1.37	2.12	0.41	0.52
		500	0.77	0.94	0.75	0.07	1.36	2.50	0.24	0.40
	5	25	3.35	3.52	2.71	0.09	0.48	0.66	0.43	0.61
		50	2.31	2.86	2.07	0.05	0.47	0.46	0.37	0.31
		100	1.70	1.96	1.53	0.03	0.21	0.43	0.21	0.30
		200	1.12	1.50	1.07	0.04	0.36	0.87	0.25	0.28
		500	0.77	1.01	0.76	0.05	0.46	1.20	0.13	0.17
			1.81	**2.08**	**1.56**	**0.07**	**0.85**	**1.32**	**0.42**	**0.49**
OVERALL			**1.91**	**2.18**	**1.58**	**0.11**	**1.04**	**1.22**	**0.53**	**0.55**

Table 3: Upper bound comparisons

The results displayed in Table 3 gain being analysed with respect to the computing times of the upper bound procedures. Detailed results, available from the authors, show that they are generally insignificant for small-size instances. For larger instances (500 items), BFD(3), L-BFD(1, 0.1, 3), and C-BFD(3) require computing times of less than 0.1 seconds. DIVE(1) and DIVE(2) require computing times of 1 second, while B-DIVE(1) and B-DIVE(2) take about 0.3 seconds. Z_{SC} is the most time consuming heuristics with a computing time that may go to the time limit of 20 seconds. We further discuss computing-time issues for Z_{SC} in Section 6.4.

Fast solutions can thus be obtained through the BFD(3) heuristics,but the quality is not very good, the overall average gap being 1.91%. L-BFD(1, 0.1, 3) is, in principle, worse than BFD(3), with a gap of 2.18%. But, if exploited to compute the C-BFD(3) heuristics, the quality improves, the gap reducing to 1.58%. The best results are yielded by Z_{SC} with an overall gap of 0.11%. Nevertheless, as mentioned before, this is also the most time consuming heuristic. A good compromise between accuracy and efficiency is offered by the diving strategies (DIVE(1), DIVE(2), B-DIVE(1), and B-DIVE(2), with average gaps varying between 0.53% and 1.22%.

6.4. Sensitivity analysis

This subsection is dedicated to the analysis of the impact on the performance of best bound procedures on a number of important problem parameters: the percentage of compulsory items, the total number of available bins. and the variability of item volumes and profits.

6.4.1. Percentage of compulsory items

We consider the instances of Class 3, where 12 large instances (500 items) taken from Class 1 and Class 2 are selected for each percentage of compulsory items set at 0%, 25%, 50%, 75%, and 100%. Table 4 display comparative results for three GBPP upper bounds:

BEST BFD: Best among the constructive heuristics BFD(3), L-BFD(1, 0.1, 3), and C-BFD(3);

Z_{SC} : Our best upper bound;

BEST DIVING: Best between the diving heuristics B-DIVE(1) and B-DIVE(2).

Table 4a displays the mean percentage gap between these upper bound and the lower bound LB_1 over the 12 instances, while Table 4b lists the corresponding computing times in seconds.

The results clearly show a trend. The most difficult instances for all the upper bounds are those where compulsory and non-compulsory items are more or less of equal quantity. All upper bounds perform better when one type of items (compulsory or non-compulsory) dominates the other. The ranking among the different upper bounds previously observed is confirmed by these results: the best constructive heuristics is extremely fast but yields worse results than the best diving method, which is outperformed by the Z_{SC}.

6.4.2. Total number of available bins

The analysis of the impact of the total number of available bins on the accuracy of the proposed bounds, we computed the relative gaps between our best upper bound Z_{SC} and the lower bounds LB_1, LB_2, and LB_3. Computations were performed on the 24 instances of Class 4. Recall that these instances were selected from Classes 1 and 2 for which we reduced the total number of available bins U by a percentage ranging from 0% up to 30%, with a step of 10%.

Figure 1 displays the gaps after 20 seconds of computing time. The behaviour illustrated by the figure is that the tighter the constraint on the total number of available bins, the more the accuracy of the gap degrades. The question then is whether this degradation follows from inaccuracies in the values of the upper bounds or from the tightness of the constraints on the bin supply. To start answering this question, we need Z_{SC} to be as close as possible to the optimal solution for most instances; 20 seconds is too short for most cases, however. We therefore let computations continue until a time limit of 1000 seconds. The corresponding gaps between Z_{SC} and the three lower bounds are displayed in Figure 2.

The trend is similar to that of Figure 1 with just a small reduction (0.3%) in the gap values. As the values of the upper bound Z_{SC} are now close or equal to the optimal ones, we conclude that the gap degradation is not due to the accuracy of Z_{SC}, but to the impact of the constrains on the total number of available bins.

% COMPULSORY ITEMS	BEST BFD % GAP	Z_{SC} % GAP	BEST DIVING % GAP
0	0.77	0.12	0.37
25	1.73	0.51	1.00
50	12.21	2.72	7.52
75	2.04	0.52	1.15
100	0.85	0.16	0.51
MEAN	**3.52**	**0.81**	**2.11**

(a)

% COMPULSORY ITEMS	BEST BFD (seconds)	Z_{SC} (seconds)	BEST DIVING (seconds)
0	< 0.01	11.37	0.39
25	< 0.01	11.94	0.55
50	< 0.01	13.95	0.42
75	< 0.01	13.41	0.33
100	< 0.01	13.42	0.25
MEAN	< **0.01**	**12.82**	**0.39**

(b)

Table 4: Impact of the percentage of compulsory items on the best upper bounds

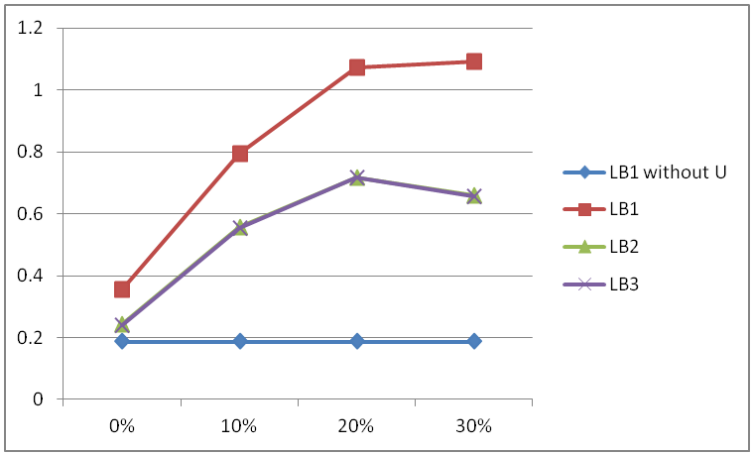


Figure 1: Class 4 gaps between Z_{SC} and LB_1, LB_2, and LB_3 for varying U with a time limit of 20 seconds

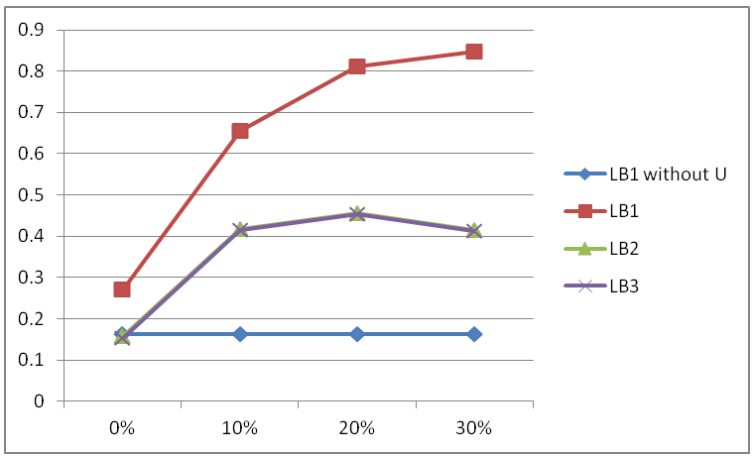


Figure 2: Class 4 gaps between Z_{SC} and LB_1, LB_2, and LB_3 for varying U with a time limit of 1000 seconds

6.4.3. Impact of item volumes and profits

The last set of experiments had a double objective. First, to inquire whether longer computation times may improve the accuracy of Z_{SC}, the best GBPP upper bound we propose. Second, to evaluate the impact of the variability in item volumes and profits on the performance of the same upper bound procedure.

Experiments were thus performed by extending the time limit of the Z_{SC} procedure to 10000 seconds. All instances with up to 200 items were solved to optimality in at most 60 seconds, independently of the parameters used to generate the instance data. The behaviour changed when the number of items was increased to 500, the time required to reach the best solutions and the rate of improvement varying with the problem characteristics.

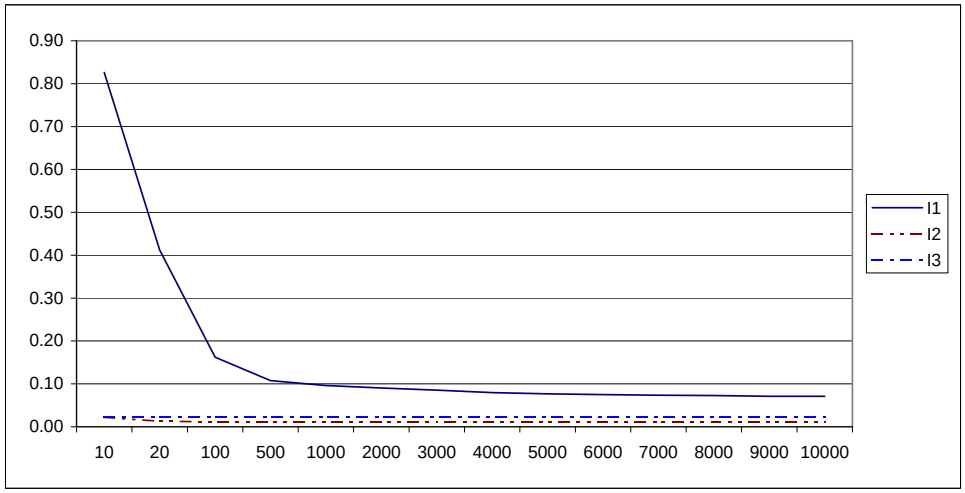
Figures 3a and 3b plot the evolution of the mean percentage gap between Z_{SC} and LB_1, when the computing time is increased. Instances are grouped by item volume (I1, I2, and I3) in Figure 3a, whilst the item profit taken from Classes 0, 1, and 2 is used to group instances in Figure 3b.

The results indicate that the most challenging instances are characterized by a large variance in item volume (I1), the method achieving very rapidly extremely good results on all other problem instances. In this case, see Figure 3a, the Z_{sc} heuristics achieves rapidly (around 100 seconds) a gap of less than 0.2%, requires four time to drop to 0.1%, and then, until the 10000 seconds time limit, continues to drop slowly to a gap of less than 0.1%. After this time limit, the convergence process slows down considerably. This behaviour is partially explained by the fact that the column generation generates a large number of patterns when there is significant variation in item volumes. Then, the general-purpose branch & bound software used has to consider a large number of variables, causing the reduction of its convergence rate. Moreover, a wide choice in terms of patterns also degrades the lower bound precision. This behaviour points to the interest of developing a particular exact exact method for the GBPP taking advantage of the peculiarities of this new class of problems.

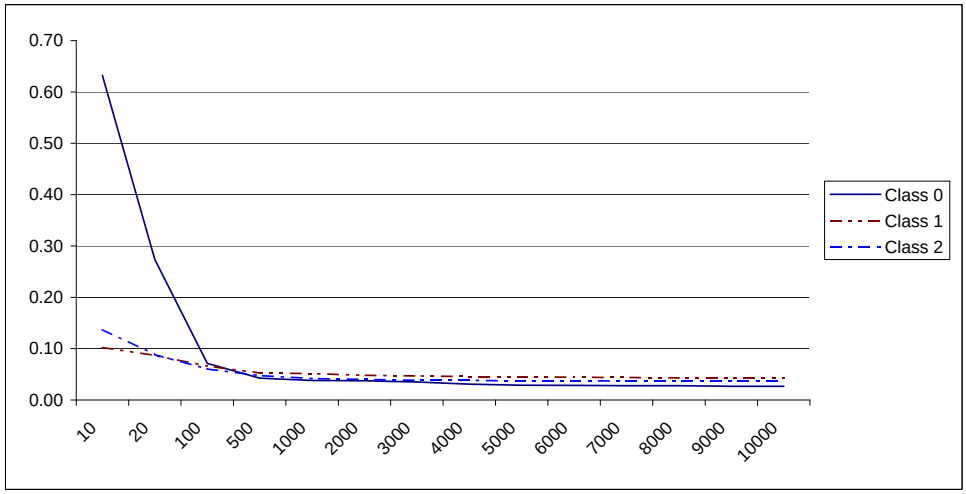
Not surprisingly, see Figure 3b, Z_{SC} is most challenged by problem instances not displaying the characteristics the bounding procedures were developed for. In our case, these instances are in Class 0 where all items are compulsory and for which the procedure requires about 300 seconds on average to reach a gap of less than 0.1%. It is very encouraging, however, to observe that, not only is the procedure performing very well on GBPP problem instances, but its behaviour is also good on instances lacking the characteristics it was designed for.

7. Conclusions

We introduced the Generalized Bin Packing Problem, a new bin packing problem of great interest to the fields of packing as well as freight transportation and logistics. The contribution of the paper to the literature is of two orders. First, it generalizes many existing packing problems by introducing a generalized formulations that jointly considers transportation costs, profits derived from orders, and constraints on order shipments. Second, it presents



(a)



(b)

Figure 3: Mean Z_{SC} - LB_1 gap versus computing time for 500-item instances

several lower and upper bound procedures, analyses them in depth, and proposes a solution method that is both very efficient computationally and offers very high-quality solutions.

In the GBPP, given two sets of compulsory and non-compulsory items characterized by volume and profit and a set of bins with given volume and cost, one aims to select the subset of profitable non-compulsory items to be loaded together with the compulsory ones into the appropriate bins in order to minimize the total net cost defined as the difference between the total cost of the selected bins and the total profit of the loaded items. We presented two formulations of the problem, based on item-to-bin assignment decisions and set covering, respectively. The set covering formulation proved to be the most interesting in algorithmic terms, and led to an efficient column generation-based lower bound method. We also presented first fit, best fit and column generation-based upper bound procedures. The paper introduced a large number of instances for this new problem. The analysis of the extensive computational results showed that the procedures proposed are very efficient and the bounds are tight.

Acknowledgments

While working on this project, the second author was the NSERC Industrial Research Chair in Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway.

This project has been partially funded by the Ministero dell'Istruzione, Università e Ricerca (MIUR) (Italian Ministry of Education, University, and Research), under the Progetto di Ricerca di Interesse Nazionale (PRIN) 2009, "Methods and Algorithms for the Logistics Optimization", and the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs, and by the partners of the Chair, CN, Rona, Alimentation Couche-Tard and the Ministry of Transportation of Québec.

References

[1] C. Alves and J. M. Valério de Carvalho. Accelerating column generation for variable sized bin-packing problems. *European Journal of Operational Research*, 183:1333–1352, 2007.

[2] A. Atamturk and M. W. P. Savelsberg. Integer programming software systems. *Annals of Operations Research*, 140:67–124, 2005.

[3] A. Mainville Cohn and C. Barnhart. The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning. In *in Proceedings of the Triennial Symposium on Transportation Analysis*, 1998.

[4] I. Correia, L. Gouveia, and F. Saldanha-da-Gama. Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, 35:2103–2113, 2008.

[5] 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 & Operations Research*, 38:1474–1482, 2011.

[6] T.G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei. New Bin Packing Fast Lower Bounds. *Computers & Operations Research*, 34(11): 3439–3457, 2007.

[7] T.G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei. Computing the Asymptotic Worst-Case of Bin Packing Lower Bounds. *European Journal of Operational Research*, 183(3):1295–1303, 2007.

[8] W.F. de la Vega and G.S. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[9] S. P. Fekete and J. Schepers. New classes of lower bounds for bin packing problems. *Mathematical Programming*, 91(1):11–31, 2001.

[10] Gurobi Optimization. *Gurobi solver 4.0 reference manual*, 2010.

[11] ILOG Inc. *IBM ILOG CPLEX v12.1 User's Manual (2009)*, 2009.

[12] J. Kang and S. Park. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147:365–372, 2003.

[13] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science*, pages 312–320. IEEE Comput. Soc. Press., 1982.

[14] H. Kellerer, U. Pferschy, and D. Pisinger, editors. *Knapsack Problems*. Springer Verlag, 2004. ISBN 3-540-40286-1.

[15] Z. Li. *Optimal Shipping Decisions in an Airfreight Forwarding Network*. PhD thesis, University of Waterloo, Ontario, Canada, 2011.

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

[17] M. Monaci. *Algorithms for packing and scheduling problems*. PhD thesis, Università di Bologna, Bologna, Italy, 2002.

[18] D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, Copenhagen, Denmark, 1995.

[19] P. Schwerin and G. Wäscher. The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, 4:377–389, 1997.

[20] S. S. Seiden, R. Van Stee, and L. Epstein. New bounds for variable-sized online bin packing. *SIAM Journal on Computing*, 32(2):455–469, 2003.

[21] K. Shintani, A. Imai, E. Nishimura, and S. Papadimitriou. The container shipping network design problem with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, 43:39–59, 2007.

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