The generalized bin packing problem

CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Generalized Bin Packing Problem

Mauro Maria Baldi
Teodor Gabriel Crainic
Guido Perboli
Roberto Tadei

July 2011

CIRRELT-2011-39

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

The Generalized Bin Packing Problem[†]

Mauro Maria Baldi[1], Teodor Gabriel Crainic[2,3,*], Guido Perboli[1,2], Roberto Tadei[1]

[1] Department of Control and Computer Engineering, Politecnicò di Torino, Corso Duca degli Abruzzi, 24 - I-10129 Torino, Italy

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. We introduce the Generalized Bin Packing Problem, a new packing problem where, given a set of items characterized by volume and profit and a set of bins with given volume and cost, one aims to select the subsets of profitable items and appropriate bins to optimize an objective function which combines the cost of the used bins and the profit derived by loading the selected items. The Generalized Bin Packing Problem thus generalizes many other packing problems, including the Bin Packing Problem and the Variable Cost and Size Bin Packing Problem, as well as the Knapsack, the Multiple Homogeneous and the Heterogeneous Knapsack Problem. We present two formulations of the problem, which are the basis for the lower bound computations, i.e. an aggregate knapsack lower bound and a column generation-based lower bound method. Upper bounds are obtained by using first fit, best fit, and column generation-based procedures. The paper also introduces new instance sets and analyzes the results of extensive computational experiments, which show that the proposed procedures are efficient and the bounds are tight.

Keywords. Generalized Bin Packing, heuristics, column generation, bounds.

Acknowledgements. While working on this project, T.G. Crainic was the Natural Sciences and Engineering Research Council of Canada (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 supported by the Ministero dell'Istruzione, Università e Ricerca (MIUR) (Italian Ministry of University and Research), under the "Progetto di Ricerca di Interesse Nazionale" (PRIN) 2007, "Optimization of Distribution Logistics", and the Natural Sciences and Engineering Research 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 Quebec.

[†]This version updates the CIRRELT-2010-21

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

1 Introduction

Packing problems consider sets of items and item-holding objects called bins, and aim to group items in such a way that they all (or a maximum number of them) fit into the minimum number of bins. The interest in packing problems follows from their own importance as a hard combinatorial optimization problem class (in all but their simplest form), as well as from the wide range of applications of theoretical and practical importance, e.g., loading, cutting, scheduling, and routing. Packing problems are particularly relevant for transportation and logistics, not only in the classical operational settings of loading items into "vehicles" (pallets, containers, trailers, trucks, rail cars, ships, and so on) and warehousing spaces, but also in numerous planning activities, from the design of plans and schedules to the participation to electronic markets.

The consideration of various rules and restrictions on the accommodation of items into bins, as well as of different item and bin attributes, has yielded several particular problem settings and a significant body of contributions (see (Wäscher et al., 2007) for a detailed survey). A number of issues and challenges remain, however. Thus, despite a number of recent efforts targeting the Variable Cost and Size Bin Packing Problem (VCSBPP) (Correia et al., 2008; Monaci, 2001; Crainic et al., 2011), there is still no general modeling framework able to address satisfactorily different variants of packing problems. The specification of objective functions particular to each problem variant appears as a major challenge in this respect. It is interesting to contrast this situation to the vehicle routing field where common structures in the objective function have yielded more general formulations and solution methods, e.g., (Cordeau et al., 2001; Pisinger and Ropke, 2007). Moreover, there are no formulations simultaneously addressing the "cost" attributes of bins and items, a problem setting of particular relevance for transportation and logistics. The objective of this paper is to contribute to address both these issues.

In this paper, we introduce a new packing problem, the *Generalized Bin Packing Problem* (*GBPP*) 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.

An important feature of the GBPP is that it generalizes many packing problems present in the literature, such as the Bin Packing Problem (BPP), the VCSBPP, as well as the Multiple Homogeneous and the Heterogeneous Knapsack Problem. 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.

From a practical point of view, the GBPP models many real-life situations coming from logistics, telecommunications, and machine scheduling. In logistics, the GBPP arises in freight shipping, when a company has to ship orders to different customers. Some of

these orders may be urgent whilst the others may wait before being shipped. This justifies the introduction of compulsory and not compulsory items, which profit can represent either a real profit but also the urgency of that item to be shipped. Other applications are in telecommunications, where capacity planning and frequency assignment problems arise, and in production scheduling, where a machine is represented by a bin and a task is represented by an item.

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 in discussing how the GBPP generalizes the packing problems mentioned above, but it is not suitable for efficient computation. It is, however, the starting point to compute the aggregate knapsack lower bound. We thus introduce a second model based on feasible loading patterns and set covering ideas. Despite requiring an exponential number of variables, this latter model facilitates the development of efficient solution methods. We thus present several procedures to compute lower and upper bounds for the GBPP and show their effectiveness through an extensive experimental phase.

The contributions of this paper are threefold. We introduce a new packing problem that is both relevant for many transportation and logistics planning problems and generalizes a wide range of packing and knapsack problems. We present two formulations for the problem, including one that yields an efficient column generation-based lower bound method, as well as first fit, best fit, and column generation-based upper bound procedures. Finally, new sets of instances specially designed for the GBPP are introduced. An extensive set of experiments shows that the proposed procedures are very efficient and the bounds are very tight.

The paper is organized as follows. Section 2 recalls the main contributions to the BPP and VCSBPP literature, the two problems immediately generalized by the GBPP . The GBPP formulations are introduced in Section 3, while lower and upper bounds are presented in Sections 4 and 5, respectively. Instance sets and computational results are presented and discussed in Section 6. We conclude in Section 7.

2 Literature Review

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

The objective of the BPP is to load all the items while minimizing the number of used bins. The problem has been extensively studied in the past decades, producing several exact and heuristic methods (Martello and Toth, 1990). The first bounds were proposed

by Martello and Toth (1990). New lower bounds were developed some ten years later by Fekete and Schepers (2001) by means of dual feasible functions. Starting from this paper, Crainic et al. (2007a,b) developed fast and more accurate lower bounds, able to reduce the optimality gap for a number of hand instances. A different approach was defined in Vanderbeck (1996), 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 (1981) and Karmarkar and Karp (1982) allowing to approximate an optimal solution within $1 + \epsilon$ for any fixed ϵ. However, these results are hardly usable in practice, due to the enormous size of the constants characterizing the polynomials. The literature thus shows that the most commonly used methods to achieve computational efficiency together with solution quality are the *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics, sometimes combined with local improvement heuristics (Schwerin and Wäscher, 2006).

The VCSBPP is a generalization of the BPP, where all the items must be loaded, but the bins can be chosen among several classes 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 were recently dedicated to the VCSBPP. Correia et al. (2008) 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. (2011) proposed tight lower and upper bounds, which could be computed within a very limited computational time, and were able to solve to optimality all the instances proposed in Correia et al. (2008). Moreover a first computational study of the sensitivity of the optimal cost with respect to the cost definition was presented. A special case of the VCSBPP is the Variable Size Bin Packing Problem (VSBPP) where the bin costs are equal to their associated volumes. The aim of VSBPP is then to minimize the wasted volume.

Monaci (2001) in his Ph.D. thesis presented a series of lower bounds and solution methods (both heuristic and exact) for the VSBPP. His exact method was able to solve to optimality most of the instances. Some 500 item-instances are still open. Kang and Park (2003) developed two greedy algorithms for another special case of the VSBPP, where the unit cost of each bin does not increase as the bin size increases, and analyzed their performances on instances with and without divisibility constraints (in the former case, on both the item and bin sizes, or on the bin sizes only). Seiden et al. (2003) proposed upper and lower bounds for the on-line version of the problem.

An integer linear formulation for the two-dimensional VSBPP was proposed by Pisinger

and Sigurd (2005), together with lower bounds obtained applying Dantzig-Wolfe decomposition and an exact branch-and-price algorithm. Alves and Valerio De Carvalho (2007) applied a column generation approach to the mono-dimensional problem and proposed a series of strategies aimed to accelerate the solution method.

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. Part of the items, which we denote *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 to take and to select the bins to load the compulsory and the selected 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.

We start this section with a formal description of the GBPP and then we propose two possible formulations of the problem. The first formulation extends to the GBPP the assignment model of the BPP Martello and Toth (1990). Even though this type of formulation is not often used in practice, we exploit it to derive a first lower bound, called LB_1. Then we propose a set covering formulation of the problem, which allows us to introduce efficient algorithms. Indeed, the latter model is the startup for a column generation procedure, which produces the lower bound LB_2. Moreover, by exploiting the patterns created during the column generation step, accurate upper bounds can also be computed.

3.1 Notation

Let $\mathcal{I}$ denote the set of items, with $n = |\mathcal{I}|$, and let w_i and p_i be the volume and the profit of item $i \in \mathcal{I}$, respectively. Define $\mathcal{I}^{\mathrm{C}} \subseteq \mathcal{I}$, the subset of items that must absolutely be loaded, and $\mathcal{I}^{\mathrm{NC}} = \mathcal{I} \setminus \mathcal{I}^{\mathrm{C}}$ the subset of non-compulsory items, some of which may be chosen if profitable. Let $\mathcal{J}$ denote the set of available bins, with $m = |\mathcal{J}|$ and let $\mathcal{T}$ 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, denote $U \leq \sum_{t \in \mathcal{T}} U_t$ the maximum number of bins of all types one can use. The item-to-bin accommodation rules of the GBPP may then be stated as:

- All items in I^{C} must be loaded;

- For all used bins, the sum of the volumes of the items loaded into the bin must be less than or equal to the volume of that bin;
- 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 U.

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 selected (if non-compulsory) and loaded into bin $j \in \mathcal{J}$, 0 otherwise.

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

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

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

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

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

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

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

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

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

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

The objective function (1) minimizes the total net cost of the operations, given by the difference between the total cost of the used bins and the total profit of the selected non-compulsory items. Let us note that the profit of the compulsory items is not included

because it corresponds to 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 settings.

Constraints (2) have the double effect of linking the usage of the bins to the accommodation of items and to limit 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, respectively. Finally, constraint (7) limits the total number of selected bins.

The model (1)-(9) is named *AM* and *R-AM* its continuous relaxation. *AM* involves a polynomial number of variables and constraints. It is not well suited, however, to efficient algorithmic developments, due to the significant solution-space symmetry of the item-to-bin assignment variables, which is typical of these compact models of packing problems.

As mentioned above, *AM* is the starting point to compute our first Lower Bound, LB_1, as an aggregate knapsack lower bound.

Furthermore, *AM* is also suitable to show how this new problem generalizes some classical packing problems.

3.3 Relationships to other packing problems

The GBPP generalizes several classical packing problems, including the BPP (Martello and Toth, 1990), which may be obtained by considering a single bin type with $C_j = 1, j \in \mathcal{J}$ and $\mathcal{I}^{\mathrm{NC}} = \emptyset$, i.e., all the items must be loaded. Constraints (4) - (6) become then redundant and the objective becomes the minimization of the number of bins characteristic of the BPP.

Allowing several bin types but still with $\mathcal{I}^{\mathrm{NC}} = \emptyset$ yields the VCSBPP, where the total cost of the selected bins $\sum_{j\in\mathcal{J}} C_j y_j$ is minimized (constraint (4) is redundant) (Monaci, 2001; Crainic et al., 2011). Notice that an equivalent formulation for the VCSBPP can be obtained by imposing $\mathcal{I}^{\mathrm{C}} = \emptyset$ 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 accommodated into it.

The GBPP can similarly generalize several knapsack problems (see Dyckhoff, 1990, for a detailed review). Since this underlines the generality of the problem we introduce in this paper, we give the main simplifications of the GBPP that yield each particular

model:

- *Knapsack*: $|\mathcal{T}| = 1$, $|\mathcal{J}| = 1$, and $\mathcal{I}^{\mathrm{C}} = \emptyset$;
- *Homogeneous Multiple Knapsack*: $|\mathcal{T}| = 1$, $|\mathcal{J}| = m$, and $\mathcal{I}^{\mathrm{C}} = \emptyset$;
- *Heterogeneous Multiple Knapsack*: $|\mathcal{T}| > 1$, $|\mathcal{J}| = m$, and $\mathcal{I}^{\mathrm{C}} = \emptyset$.

3.4 Set Covering formulation of the GBPP

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

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 respecting all dimension restrictions and accommodation rules. Let $\mathcal{K}_t = \{k\}$ be the set of all feasible loading patterns for the bin type $t \in \mathcal{T}$, and $\mathcal{K} = \bigcup_{t\in\mathcal{T}} \mathcal{K}_t$. A feasible loading pattern k is described by a vector A_k of indicator functions $a_k^i,\ k \in \mathcal{K}_t,\ t \in \mathcal{T}$, such that $a_k^i = 1$ if item $i \in \mathcal{I}$ is accommodated into pattern k of bin type t, 0 otherwise. The cost of pattern k is then the difference between the cost of the bin type and the profits of the non-compulsory items in the pattern: $c_k = C_t - \sum_{i\in\mathcal{I}^{\mathrm{NC}}} a_k^i p_i$.

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

$$\text{Minimize} \quad \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} c_k\lambda_k \tag{10}$$

$$\text{Subject to} \quad \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} a_k^i\lambda_k = 1 \quad i \in \mathcal{I}^{\mathrm{C}} \quad \text{(dual variable } \mu_i \text{ free)} \tag{11}$$

$$\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} a_k^i\lambda_k \leq 1 \quad i \in \mathcal{I}^{\mathrm{NC}} \quad \text{(dual variable } \nu_i \leq 0) \tag{12}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_k \leq U_t \quad t \in \mathcal{T} \quad \text{(dual variable } \alpha_t \leq 0) \tag{13}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_k \geq L_t \quad t \in \mathcal{T} \quad \text{(dual variable } \beta_t \geq 0) \tag{14}$$

$$\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} \lambda_k \leq U \quad \text{(dual variable } \epsilon \leq 0) \tag{15}$$

$$\lambda_k \in \{0,1\} \quad k \in \mathcal{K} \tag{16}$$

The objective function (10) minimizes the total cost of the selected bin loading patterns and, thus, of the selected bins and items. Since the feasibility of item-to-bin accommodations is guaranteed by the definition of the decision variables, constraints (2) are not required. Inequalities (11) and (12) have the same meaning as (3) and (4) in

model AM, expressing the need to load all compulsory items exactly once, while non-compulsory ones may be loaded at most once. Constraints (13), (14), and (15) replace (5) - (7) and enforce the bounds on the number of bins one is allowed to use for each type and globally, respectively.

We name SC the formulation (10)-(16) and $R\text{-}SC$ its continuous relaxation. Compared to the assignment formulation, model SC has the advantage to separate the optimality and feasibility issues, the latter being addressed in the pattern generation, not directly considered in the model that focuses on identifying the optimal combination of patterns only. This makes SC easier to generalize to, for example, the two and three dimensional versions of the GBPP.

While it is clear that models AM and SC have the same optimum, their continuous relaxations cannot be equal. In the following, we prove that the continuous relaxation of SC dominates the continuous relaxation of AM.

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

Proof. We prove the theorem by contradiction. Let us suppose that it exists an instance $\mathcal{I}'$ such that $L_{R\text{-}AM}(\mathcal{I}') > L_{R\text{-}SC}(\mathcal{I}')$. Let be $\overline{x_1}$ and $\overline{x_2}$ the optimal solutions associated to $L_{R\text{-}AM}(\mathcal{I}')$ and $L_{R\text{-}SC}(\mathcal{I}')$, respectively.

Given any solution x_2 of model $R\text{-}SC$, it is trivial to build an associated solution $x_1(x_2)$ of $R\text{-}AM$ as follows:

- for any $\lambda_k > 0$, associate the pattern k to a different bin $\overline{j}_k$ of the proper type in model $R\text{-}AM$ (recall that $k \in \mathcal{K}_t$, where t is a given bin type);
- given $\lambda_k > 0$ and the associated bin $\overline{j}_k$, put $x_{i\overline{j}_k} = a^i_k \lambda_k$.

Clearly, the objective functions of solutions x_2 and $x_1(x_2)$ have the same value by the definition of the pattern cost c_k. Moreover, all the constraints of $R\text{-}AM$ are satisfied because λ_k is a feasible bin loading pattern. Let $\overline{x_2}$ be the optimal solution associated to $L_{R\text{-}SC}(\mathcal{I}')$. Then, by the previously described rule, we can build a feasible solution $x'_1(\overline{x_2})$ of $R\text{-}AM$ with objective function $L_{R\text{-}SC}(\mathcal{I}') < L_{R\text{-}AM}(\mathcal{I}')$. This implies that $\overline{x_1}$ is not optimal, which contradicts the hypothesis. □

The set covering formulation requires a potentially exponential number of variables, however. We thus use column generation to derive another lower bound to the GBPP. We call it LB_2.

4 Lower Bounds

In this section, we introduce two lower bounds that can be computed starting from each formulation of the problem. The assignment model is the basis for a lower bound which can be calculated by solving an aggregate knapsack problem (see subsection 4.1).

The second lower bound is derived from the set-covering formulation and it 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 subsection 4.2).

4.1 Lower Bound through Aggregate Knapsack Problem

The main idea is to consider the assignment model presented in subsection 3.2 and derive an Aggregate Knapsack Problem.

We aggregate constraints (2) into a unique inequation by summing them up. 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, the variables x_{ij} can be reduced to x_i, which state wether item i is put into the aggregate knapsack, which may be thought as a unique large bin with volume equal to the sum of all bin volumes. Therefore, we have:

$$\sum_{i \in \mathcal{I}^C} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j \tag{17}$$

The lower bound LB_1 can be found by solving the following Aggregate Knapsack Problem:

$$\text{Minimize} \quad \sum_{j \in \mathcal{J}} C_j y_j - \sum_{i \in \mathcal{I}^{\mathrm{NC}}} p_i x_i \tag{18}$$

$$\text{Subject to} \quad \sum_{i \in \mathcal{I}^C} w_i + \sum_{i \in \mathcal{I}^{NC}} w_i x_i \leq \sum_{j \in \mathcal{J}} W_j y_j \tag{19}$$

$$y_j \in \{0,1\}, \qquad j \in \mathcal{J} \tag{20}$$

$$x_i \in \{0,1\}, \qquad i \in \mathcal{I} \tag{21}$$

Note that, when all the items are compulsory, we return to the VCSBPP and the aggregate knapsack problem reduces to:

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

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

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

which is the first model exploited in Crainic et al. (2011) to compute lower bounds to the VCSBPP. Therefore, when only compulsory items are present, one can use bounds from the literature.

4.2 Lower Bound through Column Generation

This lower bound, denoted as LB_2, is computed from the linear relaxation of the set covering formulation 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 (Alves and Valerio De Carvalho, 2007; Vanderbeck, 1996)).

The general column generation approach applied to the GBPP consists in starting with a relatively small set of feasible patterns $\mathcal{P}$ corresponding to a restricted GBPP that we name R-GBPP. One then first solves the linear relaxation of the set covering formulation SC of R-GBPP, and then attempts to generate new feasible patterns with negative reduced cost. If successful, these are added to $\mathcal{P}$ and the procedure is restarted; otherwise, one has identified the optimal solution of the R-SC and the procedure stops with a lower bound to the GBPP. Formally,

1. Find an initial feasible solution of the GBPP and the corresponding set $\mathcal{P}$;

2. Solve the linear relaxation *R-SC* of the R-GBPP and let $L_{\text{R-GBPP}}$ be its optimal solution;
3. For each bin type $t \in \mathcal{T}$
   (a) Find the non-basic pattern variable $\lambda_{\overline{k}}$ of $L_{\text{R-GBPP}}$, $\overline{k} \in \mathcal{K}_t$, with the smallest reduced cost $r_{\overline{k}}$ among all non-basic pattern variables for type t;
   (b) If $r_{\overline{k}} < 0$, $\mathcal{P} = \mathcal{P} \cup \{\lambda_{\overline{k}}\}$;
   (c) Continue to the next bin type (go to 3a);
4. If $r_{\overline{k}} \geq 0$ for all bin types t, then stop with $L_{\text{R-GBPP}}$ as the set covering lower bound to the GBPP, otherwise, go to 2.

The main issue is how to find new negative reduced-cost feasible patterns. Consider the dual variables associated to the constraints of the continuous relaxation of the R-GBPP set covering formulation, then the reduced cost r_k of a given pattern variable k for a bin of type t is $c_k - [\mu\ \nu\ \alpha\ \beta\ \epsilon]^T A_k$. We expand this expression as follows:

$$\begin{aligned} r_k &= C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} p_i\, a_k^i - \left[\mu^{\mathbf{T}}\ \nu^{\mathbf{T}}\ \alpha^{\mathbf{T}}\ \beta^{\mathbf{T}}\ \epsilon\right] A_k \\ &= C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} p_i\, a_k^i - \sum_{i \in \mathcal{I}^{\text{C}}} \mu_i\, a_k^i - \sum_{i \in \mathcal{I}^{\text{NC}}} \nu_i\, a_k^i - \alpha_t - \beta_t - \epsilon \\ &= C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} (p_i + \nu_i)\, a_k^i - \sum_{i \in \mathcal{I}^{\text{C}}} \mu_i\, a_k^i - \alpha_t - \beta_t - \epsilon \end{aligned} \tag{22}$$

We now define a column generation subproblem which, given a bin of type $t \in \mathcal{T}$, finds the non-basic pattern with the minimum reduced cost. Notice that the vector A_k defining a not-yet-generated pattern $k \in \mathcal{K}_t$ of bin type $t \in \mathcal{T}$ is not known but may be expressed in terms of item-to-bin assignment variables x_i equal to 1 if item $i \in \mathcal{I}$ belongs to the pattern, 0 otherwise (for simplicity of notation, we drop the k index). Since the dual variables α_t, β_t, and ϵ, as well as the bin cost C_t, are constant for a given type $t \in \mathcal{T}$, the reduced cost of the yet-unknown pattern one desires to minimize becomes

$$\begin{aligned} r_k &= \left\{ C_t - \sum_{i \in \mathcal{I}^{\text{NC}}} (p_i + \nu_i)\, x_i - \sum_{i \in \mathcal{I}^{\text{C}}} \mu_i\, x_i - \alpha_t - \beta_t - \epsilon \right\} \\ &= \left\{ - \sum_{i \in \mathcal{I}^{\text{NC}}} (p_i + \nu_i)\, x_i - \sum_{i \in \mathcal{I}^{\text{C}}} \mu_i\, x_i \right\} \end{aligned} \tag{23}$$

Finding the feasible pattern with minimum reduced cost for bin type $t \in \mathcal{T}$ then becomes a knapsack problem:

$$\begin{aligned} \text{Maximize} \quad & \left\{ \sum_{i \in \mathcal{I}^{\mathrm{NC}}} (p_i + \nu_i)\, x_i + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i\, x_i \right\} \\ \text{Subject to:} \quad & \sum_{i \in \mathcal{I}} w_i x_i \leq W_t \qquad t \in \mathcal{T} \\ & x_i \in \{0,\ 1\} \qquad i \in \mathcal{I} \end{aligned}$$

Any feasible solution may be used to initialize the procedure, including the trivial solution obtained loading each compulsory item into a different bin. More accurate heuristics are presented in Section 5. The procedure adds at most $|\mathcal{T}|$ columns to $\mathcal{P}$ at every iteration (in Step 3), one for each bin type, yielding a feasible loading pattern with $r_{\bar{k}}$ negative.

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 present several upper bounds for the GBPP. Upper bounds can be found through constructive heuristics. They are based on the well-known *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics for the BPP, and they are introduced in subsection 5.1, together with the lower bound based on the constructive heuristics. Next, in subsection 5.2, we show how to derive upper bounds from the column generation procedure by solving an integer problem based on the created columns (or patterns), or by iteratively rounding those columns along some given stategies. The latter procedure is known as diving.

5.1 Constructive heuristics

We propose heuristics to load items into bins derived from the *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) heuristics for the BPP. Briefly, starting with an initial sorting of the items in non-increasing order of their volumes, the FFD loads items one after the other into the first bin where they fit, while BFD attempts to load each item into the "best" bin where it can be accommodated, usually the bin which, after loading the item, has the minimum *free volume*, defined as the bin volume minus the sum of the volumes of the items it contains. Both heuristics create a new bin when an

item cannot be accommodated into existing ones. Despite their simplicity, the FFD and BFD heuristics offer good performances for the BPP (Martello and Toth, 1990).

Whilst in the classical BPP items are ordered by non-increasing volumes, here, due to different items and bins attributes, many sortings may take place.

Note that the aggregate knapsack lower bound introduced in 4.1 yields the sets of bins and items making up the lower bound itself. The key to derive an upper bound is to consider two percentages δ_j and δ_i of, respectively, those bins and items which have been selected in the lower bound computation. Of course, chosen bins will be taken into account as empty. Therefore one gets two lists, one for the selected bins and one for the selected items at their top plus the remaining bins and items following. When a constructive heuristic is performed using the two mentioned lists, a new upper bound is available. Since, in this case, information of the aggregate knapsack lower bound is involved, we talk about Lower Bound-based FFD and Lower Bound-based BFD, shortened respectively as L-FFD and L-BFD.

Applying the first or best fit idea to the GBPP presents a number of challenges, however, which follow from the heterogeneity of the bins and the interplay among bin cost, non-compulsory item profit, and the volumes of the respective bins and items. First, one has to sort not only items but also bins. Furthermore, while volumes are, of course, important, costs and profits could influence the order as well. Several sorting strategies are thus possible and we examine four, all having in common the compulsory items sorted in non-increasing values of their volumes and placed at the top of the item list. The four sorting strategies are:

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

When performing a lower bound-based constructive heuristic, we first have to compute the aggregate knapsack problem, i.e. LB_1. We take into account those items and those bins which have been selected when solving the aggregate knapsack problem and we select two portions among them, given by percentages (to be set during the calibration step to be, introduced next) δ_i and δ_j, respectively. Such items and bins will appear at the top of their respective lists and will be sorted according to one among the four above sorting strategies. The remaining items and bins (i.e. those which have not been taken into account when solving the aggregate knapsack problem) will follow in their respective lists and they will be sorted according to the selected ordering strategy.

Given ordered sets of bins and items, the FFD and BFD heuristics proceed according to the standard sequence, the former loading each item into the first already-selected bin with sufficient available volume, while the latter selects the existing bin maximizing the MERIT FUNCTION$(i, b) = \frac{1}{r_b+1}$, where r_b stands for the resulting bin free volume as defined above. In both cases, the next bin in the list is selected when a new bin is required to load a compulsory item.

Algorithm 1 The PROFITABLE(item i, container b) Heuristic for New Bin Selection

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

Major differences with the FFD and BFD heuristics for the BPP are that 1) one must chose a profitable subset of non-compulsory items to load, that is, items for which their total profits decrease as much as possible the total cost of all the selected bins, and 2) one must choose the bin to add to the solution given its particular cost and volume. Consider a non-compulsory item $i \in \mathcal{I}^{\mathrm{NC}}$ that cannot be loaded into any partially loaded container and a new one should be selected. When the profit of i and those of the remaining items in the list are so low that their sum does not exceed the cost of the new container, then it is better not to select item i. To determine whether this is the case, it is sufficient to solve a knapsack problem considering the not yet loaded items and each bin type such that at least one bin is still available. To avoid the extra computational burden, which could be significant, we take advantage of the ordering of the bins and use Algorithm 1 to compute an approximation of the value of selecting bin b for item i.

Another issue one needs to address is the behavior of the procedures towards the end of the item list, when there might not be sufficient items left to fill up the selected bin, even though it offers a good cost/volume ratio measure. A bin with a worst ratio but an absolute smaller cost might be appropriate in this case, and we include a post-processing

Algorithm 2 The BFD and FFD Upper Bound Heuristics

Sort the bins and the items (compulsory items first), and let *SCL* and *SIL* be the resulting ordered lists, respectively.
Let $\mathcal{S}$ be the set of the selected containers.
for all $i \in SIL$ **do**
  Identify the bin $b \in \mathcal{S}$ into which item i can be loaded:

- BFD: the best, i.e., the bin b maximizing the MERIT FUNCTION(i, b);
- FFD: the first with sufficient empty space to accommodate i.

  if b does not exist **then**
    if $i \in I^C$ **then**
      Identify the first container $b' \in SCL \setminus \mathcal{S}$ such that $w_i \leq W_{b'}$.
    else
      Identify the container $b' \in SCL \setminus \mathcal{S}$ such that PROFITABLE(i, b') returns TRUE.
    end if
    $b = b'$, if b' exists.
  end if
  If b exists, load i into b, reject item i, otherwise.
end for
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

phase that attempts to improve the solution by evaluating such possible bin swaps. The procedure iteratively examines each selected bin $j \in \mathcal{S}$ with a loaded volume U_j defined as the sum of the volumes of the items assigned to it. Then, if an unused bin $k \in \mathcal{J}$ exists such that $W_k \geq U_j$ and $C_k < C_j$, the items from bin j are transferred to bin k and bin j is discarded. Algorithm 2 displays the FFD and BFD heuristics developed for the GBPP.

Algorithm 3 The DIVING Heuristic

Let $\overline{x}$ be the optimal solution of the continuous relaxation of R-GBPP
while $\overline{x}$ is not integral **do**
  Select a non-integer variable λ_k and set $\lambda_k = 1$;
  Re-optimize R-GBPP.
end while

Finally, a few words about infeasibility are worthwhile. Indeed, when the number of available bins is too limited, some compulsory items may be discarded. In such situations the constructive heuristics fail and the solution found so far is infeasible. However infeasibility can be treated introducing extra bins. In a real logistics context, indeed, when a shipping company ends its available containers and needs a new one, this can be bought or rented or borrowed from partners or from other companies. This situation can be modeled by creating an extra bin type which availability is in principle infinite but which cost is very high. Such a high cost represents a penalty, discouraging the shipping company in using extra containers (in fact these should be used only when no ordinary ones are available).

5.2 Column Generation-based Heuristics

We present two approaches for computing upper bounds starting from the column generation-based solution to the relaxation *R-SC* of the set covering formulation *SC*.

The first approach is to solve exactly the MIP formulation *SC*, by branch-&-bound, for example, considering only 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.

The second approach is based on *diving*, a well-known method for finding good quality integer solutions from continuous solutions to the respective relaxations (Atamturk and Savelsbergh, 2005). The working principle is to iteratively round up variables and re-optimize the continuous relaxation. Adapted to the GBPP, the diving heuristic mechanism is displayed in Algorithm 3.

The heuristic assumes that the optimal bin loading patterns of the GBPP are in the restricted set corresponding to the R-GBPP, and slightly and iteratively perturbs the optimal continuous solution by fixing patterns to push toward an integer solution. The key feature is how to choose the variable to be fixed. Preliminary experiments showed that criteria based on the contributions of the items making up the pattern to its marginal cost yield superior performances. Two strategies, selecting the pattern variable with non-integral value maximizing expression (24) or (25), make up the two diving heuristics included in the final comparative experiments of Section 6:

$$\sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a_k^i + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a_k^i \tag{24}$$

$$(1 - \lambda_k) \left(\sum_{i \in \mathcal{I}^{\mathrm{NC}}} \nu_i \, a_k^i + \sum_{i \in \mathcal{I}^{\mathrm{C}}} \mu_i \, a_k^i \right) \tag{25}$$

6 Computational results

The goal of the numerical experiments is to explore the performance of the proposed lower bounds and heuristics.

The algorithms were coded in C++ and the models implemented with CPLEX Concert Technology. Z_{SC} was computed using Gurobi 4.0, due to its efficiency in finding good feasible solutions within a limited computational time of 20 seconds. Experiments were conducted on a Pentium IV 3.0 Ghz workstation with 4 Gb of RAM.

No instances are present in the literature for the GBPP. We generated 900 new instances, partially based on those for the VSBPP and the BPP (Monaci, 2001; Vanderbeck, 1996; Correia et al., 2008; Crainic et al., 2011).

In particular, we started from Monaci's instances (Monaci, 2001) (denoted as Class 0 in our instances) and we extended them to create two new instance sets, named Class 1 and Class 2. We chose Monaci's instances because they are more challenging than Correia's ones (Correia et al., 2008), as shown in Crainic et al. (2011).

Ten instances were randomly generated for each combination of number of items, item profit, item volume, and bin types defined as follows:

- Number of items: 25, 50, 100, 200, and 500
- Item profit:
  Class 0: all the items are compulsory

Class 1: $p_i \in \lceil U(0.5, 3)w_i \rceil$, where U stands for the uniform distribution; no compulsory items present
Class 2: $p_i \in \lceil U(0.5, 4)w_i \rceil$ and no compulsory items

- Item volume:

  I1: [1, 100]

  I2: [20, 100]

  I3: [50, 100]

- Number of bin types:
  A: three types of bin, with volumes 100, 120, and 150, respectively, and costs equal to the volumes
  B: five types of bin, with volumes 60, 80, 100, 120, and 150, respectively, and costs equal to the volumes.

We then selected 12 large size instances (500 items) from Class 1 and Class 2 with a representative mix of characteristics in terms of item volumes, item profits, and bin types and we built five instance sets with 0%, 25%, 50%, 75%, and 100% of the items which are compulsory, by randomly selecting items to be compulsory. The resulting instance set is named Class 3 and is made up by 60 instances (twelve for each percentage of compulsory items).

We first examine the general performance of the proposed lower bounds and heuristics. We then examine more in detail the impact on this performance of the relative distribution of the number of compulsory and non-compulsory items, as well as of the number of items.

Before showing the tables we would like to spend a few words about time considerations that we gathered while performing computational tests. The fastest solution is given by the constructive heuristics which time is negligible but they yield, on average, a worse solution if compared to the other mentioned methods. LB_1 took, on average, 1.2 seconds, while the column generation produced, on average, 424.16 patterns. The Z_{SC} upper bound has been solved within a time limit of 20 seconds and diving heuristics took, on average, 0.22 seconds. However one must also take into account the column generation time because it would be impossible to perform the diving heuristics without generating any pattern.

Table 1 shows lower bound results. In particular, column 1 shows the class type, column 2 the number of bin types, column 3 the number of items, columns 4 and 5 the gap and the optimum for LB_1, columns 6 and 7 the gap and the optimum for LB_2, and finally columns 8 and 9 the gap and the optimum for LB_3. Each row of Table 1 gives the results of 30 instances (3 item volume types times 10 random repetitions).

Note that, since Class 0 instances are those by Monaci Monaci (2001), we can compute gaps by referring to the Monaci's optima when available, or to the best lower bounds between those by Monaci (2001) and those by Crainic et al. (2011) (both optima and lower bounds are below indicated by z^*). Therefore, the gap for any instance I of Class 0 is computed as

$$\left| \frac{LB_x(I) - z^*(I)}{z^*(I)} \right| \times 100$$

where subscript x stands for 1, 2, or 3.

For the remaining classes, gaps are computed referring to the best lower bound, i.e. LB_3.

Table 1 shows very promising results: the overall gap is quite tight (0.07 %) and we close almost half of the created instances (415 over 900).

In Table 2 we present the constructive heuristics results. In particular, column 1 shows the class type, column 2 the number of bin types, column 3 the number of items, columns 4 to 7 the FFD gaps following the mentioned bin and item sorting rules, and finally columns 8 to 11 the BFD gaps following the mentioned bin and item sorting rules. The meaning of the rows is the same of those of Table 1.

Table 2 shows that, on average, BFD offers better results than FFD when applying the same sorting rules. Furthermore we see that, always on average, BFD 3 is the best performing constructive heuristic.

In the following two tables we compare lower bound-based constructive heuristics to simple constructive heuristics. Since, as seen in Table 2, BFD outperforms, in practice, FFD, we limit our analysis to BFD heuristics only. As seen at the end of subsection 5.1, the sorting of bins and items depends on the percentages δ_j and δ_i of the selected bins and items after applying LB_1. The parameter calibration has been done by performing all the L-BFD 3s (we limited to only consider the best constructive heuristic on average, i.e. BFD 3) on some selected instances, varying both δ_j and δ_i from 0.1 to 1 with a step of 0.1. This means that, for each instance, we performed 100 L-BFD 3s. In Table 3 we give the mean gap for every combination of the two parameters δ_j and δ_i.

The best results are found by setting $\delta_j = 0.1$ and $\delta_i = 1$. Let us note that, again, this calibration does not guarantee the best results for every instance but on average. As an alternative a good range is given by varying δ_j and δ_i between 0.1 and 0.3. In Table 4 we report such a comparison. More in detail, in column 4 the BFD 3 results are given. Column 5 shows the L-BFD 3 results with $\delta_j = 0.1$ and $\delta_i = 1$. Column 6 shows the so called *composite* BFD 3 which is given by taking, for each instance, the best (i.e. the minimum) between BFD 3 and all the L-BFD 3s, obtained by varying δ_j and δ_i between 0.1 and 0.3. The meaning of the rows is the same of those of Table 1. The

			LB_1		LB_2		LB_3	
CLASS	**BINS**	**ITEMS**	**GAP**	**OPT**	**GAP**	**OPT**	**GAP**	**OPT**
0	3	25	1.16	10	0.26	13	0.13	21
		50	0.62	12	0.17	5	0.09	16
		100	0.50	16	0.06	8	0.03	22
		200	0.31	19	0.04	6	0.02	24
		500	0.31	20	0.02	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
			0.52	**160**	**0.10**	**75**	**0.05**	**217**
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**	**0.12**	**81**	**0.10**	**99**
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**	**0.09**	**72**	**0.07**	**99**
OVERALL			**0.61**	**223**	**0.10**	**228**	**0.07**	**415**

Table 1: Lower bounds results

CLASS	BINS	ITEMS	FFD 1	FFD 2	FFD 3	FFD 4	BFD 1	BFD 2	BFD 3	BFD 4
		25	12.65	12.65	3.55	3.55	12.65	12.65	3.33	3.33
		50	13.15	13.15	2.35	2.35	13.15	13.15	2.27	2.27
	3	100	12.17	12.17	1.63	1.63	12.17	12.17	1.59	1.59
		200	10.26	10.26	1.26	1.26	10.26	10.26	1.23	1.23
0		500	8.96	8.96	1.07	1.07	8.96	8.96	1.06	1.06
		25	10.35	10.35	1.80	1.80	10.35	10.35	1.80	1.80
		50	11.54	11.54	1.74	1.74	11.54	11.54	1.73	1.73
	5	100	10.61	10.61	1.25	1.25	10.61	10.61	1.24	1.24
		200	10.82	10.82	0.82	0.82	10.82	10.82	0.81	0.81
		500	10.44	10.44	0.65	0.65	10.44	10.44	0.63	0.63
			11.09	**11.09**	**1.61**	**1.61**	**11.09**	**11.09**	**1.57**	**1.57**
		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
	3	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
1		500	9.83	13.44	0.94	6.58	9.81	13.46	0.94	6.58
		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
	5	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**
		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
	3	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
2		500	7.37	8.81	0.77	3.30	7.33	8.82	0.77	3.29
		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
	5	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			**10.02**	**12.13**	**1.93**	**4.18**	**9.99**	**12.08**	**1.89**	**4.14**

Table 2: Constructive heuristics results

		δ_i									
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
δ_j	**0.1**	2.66	2.74	2.84	2.93	2.95	2.91	2.87	2.77	2.59	2.30
	0.2	2.90	3.06	3.21	3.27	3.29	3.28	3.34	3.21	3.05	2.77
	0.3	3.25	3.29	3.44	3.61	3.62	3.65	3.63	3.55	3.43	3.18
	0.4	3.60	3.58	3.60	3.84	3.87	3.90	3.91	3.81	3.69	3.46
	0.5	3.86	3.89	3.92	4.02	4.10	4.16	4.21	4.06	3.92	3.65
	0.6	4.13	4.16	4.22	4.27	4.34	4.40	4.45	4.38	4.21	3.90
	0.7	4.40	4.45	4.52	4.54	4.54	4.63	4.69	4.66	4.57	4.23
	0.8	4.70	4.73	4.75	4.80	4.82	4.87	4.89	4.87	4.80	4.54
	0.9	4.91	5.01	5.00	5.02	5.11	5.10	5.09	5.11	5.00	4.76
	1	5.09	5.18	5.19	5.25	5.27	5.25	5.29	5.32	5.16	4.98

Table 3: Parameter calibration

results show that no heuristic (BFD 3 and L-BFD 3) dominates the other, although, on average, BFD 3 performs better than L-BFD 3. The mean gap is around 2% which can be, however, decreased if one considers the best solution between the two heuristics, i.e. C-BFD 3.

Table 5 shows comparisons among all the upper bound techniques discussed in this paper. As before, gaps are computed with respect to Monaci's optima (when available) for Class 0 instances and with respect to LB_3 for Classes 1 and 2. In particular, column 1 shows the class type; column 2 the number of bin types; column 3 the number of items; column 4 the Z_{SC} gap; columns 5 and 6 the diving 1 and diving 2 gaps respectively, and column 7 the gap when taking the best value between diving 1 and BFD 3. In fact, when computing diving, we feed the column generation procedure with BFD 3 which is, in principle, the best constructive heuristic among the others. Therefore, since it is a feasible solution and the diving has been computed, we consider the minimum between diving 1 and BFD 3. A similar issue holds for diving 2 and BFD 3 which gap is reported in column 8. Column 9 shows the best constructive heuristic gap and finally column 10 shows the composite lower bound-based BFD 3 varying both bin and item percentages between 0.1 and 0.3. The meaning of the rows is the same of those of Table 1.

In Table 5 the overall Z_{SC} gap is quite tight: 0.09%. This means that the generated columns are very good in two senses. In the first sense, because the associated IP problem (which yields the upper bound Z_{SC}) finds an optimal solution or a feasible solution close to the optimum. In the second sense, because LB_2 is a very good lower bound. Furthermore, this holds for all the involved classes, not only for Class 0. Table 5 shows also solution quality in terms of the available computational time. Referring to the previously introduced time considerations, if the available computational time is short, immediate solutions can be found by applying C-BFD 3 with a mean gap of 1.57%. Vice versa if the available computational time is larger, the column generation procedure with the Z_{SC} computation can be used. Otherwise a compromise between solution quality and time can be met by applying diving after the column generation. Depending on the used diving strategy results may vary with a gap from 0.54% to 1.21%. In any case all gaps are less than 2%.

We finally examine Class 3, i.e. the variation in the solution quality offered by the best heuristic proposed when the cardinality of the compulsory items set I^C varies.

We applied the BFD constructive heuristic, the Z_{SC} heuristic with a time limit of 20 seconds, and the diving heuristics. Table 6a displays the average gaps (%) over the 12 instances with respect to the column generation lower bound for the three heuristics, where the best result among the different versions was used for the BFD and diving heuristics. Table 6b displays the corresponding computational measures expressed in seconds.

CLASS	BINS	ITEMS	BFD 3	L-BFD 3	C-BFD 3
0	3	25	3.33	3.22	1.80
		50	2.27	2.46	1.75
		100	1.59	1.50	1.12
		200	1.23	1.33	1.01
		500	1.06	0.84	0.67
	5	25	1.80	2.36	1.62
		50	1.73	2.51	1.63
		100	1.24	1.82	1.15
		200	0.81	1.56	0.81
		500	0.63	1.25	0.63
			1.57	**1.88**	**1.22**
1	3	25	3.96	3.85	2.95
		50	3.20	2.98	2.46
		100	2.16	2.37	1.98
		200	1.45	1.77	1.37
		500	0.94	1.14	0.92
	5	25	3.33	3.67	2.42
		50	3.46	3.85	2.92
		100	1.97	2.47	1.83
		200	1.49	1.92	1.45
		500	1.00	1.21	0.97
			2.30	**2.52**	**1.93**
2	3	25	3.17	3.10	2.09
		50	2.32	2.70	2.09
		100	1.41	1.67	1.32
		200	1.20	1.49	1.15
		500	0.77	0.94	0.75
	5	25	3.35	3.52	2.71
		50	2.31	2.86	2.07
		100	1.70	1.96	1.53
		200	1.12	1.50	1.07
		500	0.77	1.01	0.76
			1.81	**2.08**	**1.56**
OVERALL			**1.89**	**2.16**	**1.57**

Table 4: Constructive heuristics best results

CLASS	BINS	ITEMS	Z_{SC}	DIVING 1	DIVING 2	BEST DIVING 1	BEST DIVING 2	BEST BFD	C-BFD 3
0	3	25	0.13	1.82	1.89	1.18	1.41	3.33	1.80
		50	0.10	1.19	1.30	0.85	0.96	2.27	1.75
		100	0.13	1.81	1.42	0.81	0.62	1.59	1.12
		200	0.16	1.73	1.13	0.58	0.49	1.23	1.01
		500	0.30	2.11	1.03	0.58	0.50	1.06	0.67
	5	25	0.00	1.14	0.74	0.55	0.47	1.80	1.62
		50	0.01	0.75	0.50	0.59	0.47	1.73	1.63
		100	0.01	0.57	0.43	0.41	0.37	1.24	1.15
		200	0.05	0.63	0.45	0.33	0.28	0.81	0.81
		500	0.05	0.76	0.41	0.16	0.13	0.63	0.63
			0.10	**1.25**	**0.93**	**0.60**	**0.57**	**1.57**	**1.22**
1	3	25	0.19	1.30	1.42	0.87	0.89	3.71	2.95
		50	0.15	1.05	0.94	0.88	0.82	3.19	2.46
		100	0.11	0.81	2.18	0.66	0.68	2.16	1.98
		200	0.12	1.57	2.28	0.55	0.61	1.45	1.37
		500	0.10	1.89	2.79	0.40	0.55	0.94	0.92
	5	25	0.12	0.74	0.76	0.55	0.58	3.26	2.42
		50	0.09	0.37	0.52	0.37	0.52	3.46	2.92
		100	0.03	0.53	0.52	0.35	0.36	1.97	1.83
		200	0.02	0.55	1.11	0.28	0.32	1.49	1.45
		500	0.07	0.73	1.18	0.15	0.18	1.00	0.97
			0.10	**0.95**	**1.37**	**0.51**	**0.55**	**2.26**	**1.93**
2	3	25	0.16	1.99	2.07	1.05	0.95	2.86	2.09
		50	0.10	0.83	1.25	0.77	0.84	2.31	2.09
		100	0.06	0.98	1.64	0.38	0.48	1.41	1.32
		200	0.10	1.37	2.12	0.41	0.52	1.20	1.15
		500	0.07	1.36	2.50	0.24	0.40	0.77	0.75
	5	25	0.09	0.48	0.66	0.43	0.61	2.68	2.71
		50	0.05	0.47	0.46	0.37	0.31	2.17	2.07
		100	0.03	0.21	0.43	0.21	0.30	1.70	1.53
		200	0.04	0.36	0.87	0.25	0.28	1.12	1.07
		500	0.05	0.46	1.20	0.13	0.17	0.77	0.76
			0.07	**0.85**	**1.32**	**0.42**	**0.49**	**1.70**	**1.56**
OVERALL			**0.09**	**1.02**	**1.21**	**0.51**	**0.54**	**1.84**	**1.57**

Table 5: Upper bound comparisons

The results clearly show a trend. The most challenging instances for all the heuristics and for which these yield the poorest quality results are those where compulsory and non-compulsory items are more or less of the same quantity. All heuristics perform very well when one type of items dominates, with their performance which is degrading with the decrease in domination. The ranking observed previously is confirmed in this study: the constructive heuristic is extremely fast but yields worse results than the diving methods, which are outperformed by the Z_{SC} heuristic, i.e., solving the branch-&-bound of the set covering formulation starting from the patterns generated by the column generation lower bound process.

It is worth noting that the performance of the Z_{SC} heuristic is very satisfactory even with respect to the most difficult instances. Moreover, it offers a very good global performance, with an average gap of 0.8% and an average computational effort of less than 13 seconds.

I^C (%)	BEST BFD	Z_{SC}	BEST DIVING
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	0.84	0.15	0.50
MEAN	**3.52**	**0.80**	**2.11**

(a)

I^C (%)	BEST BFD	Z_{SC}	BEST DIVING
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)

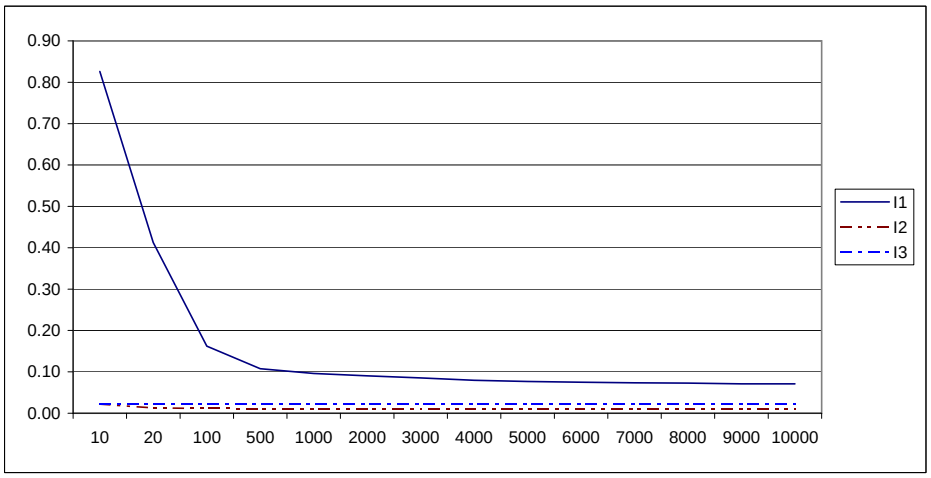
Table 6: Performance of best heuristics with different cardinality of I^C

The experimental results discussed so far point to the Z_{SC} heuristic as the upper bound procedure offering the best performance in terms of solution quality. Because the procedure is based on solving a branch-&-bound algorithm for the MIP model SC, the heuristic may also serve as a good proxy to examine the behavior of exact methods in addressing the GBPP, as well as the impact of problem size on algorithmic efficiency.

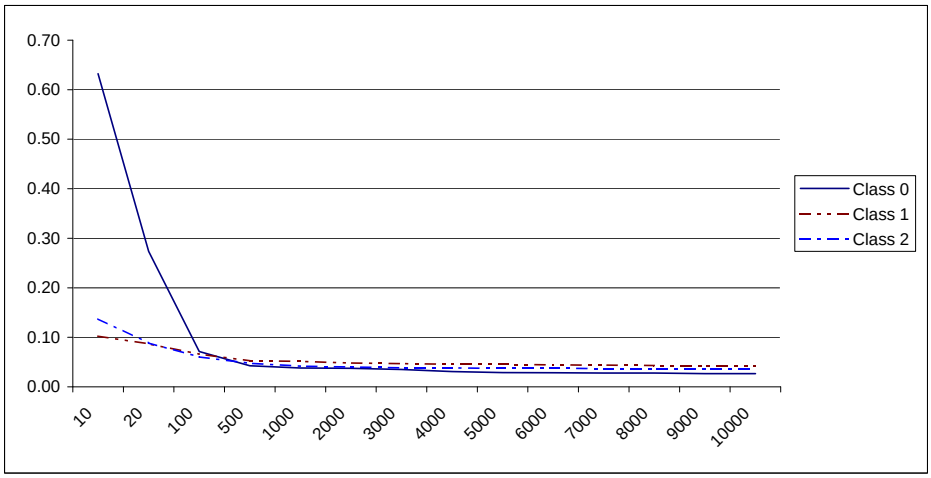
A new set of experiments was thus performed by extending the time limit of the Z_{SC} heuristic to 10000 seconds. All instances with up to 200 items were solved to optimality in at most 60 seconds, independently of the parameters used to generate the instance

data. The behavior drastically changed when the number of items was increased to 500.

Figures 1a and 1b plot the evolution of the average gap (in %) between the column generation-based lower bound and the best solution of the Z_{SC} heuristic as the computational effort, in seconds, is increased. Instances are grouped by item size (I1, I2, and I3) in Figure 1a, while the item profit (Class 0, Class 1, and Class 2) are used to group instances in Figure 1b.



(a)



(b)

Figure 1: Computational effort versus quality gap of Z_{SC} heuristic for 500-item instances

The results show how the most challenging instances are characterized by a wider variance in item sizes (item volume I1). In fact, Z_{sc} requires about 4000 seconds to converge with a gap of less than 0.1% from the lower bound. After this time limit, the convergence process slows down considerably. This is partially explained by the fact that

the column generation generates a larger number of patterns when the item volumes are different. Thus, the commercial Branch & Bound used to obtain Z_{sc} has to consider a considerably larger number of variables, causing the reduction of the convergence rate. Moreover, a larger choice in terms of patterns contributes to degrade the lower bound precision as well. This contributes to the interest of developing an ad-hoc exact method for the GBPP taking advantage of the peculiar characteristics of this new class of problems.

Not surprisingly, Z_{SC} is most challenging when all items are compulsory, i.e., belonging to Class 0, as it requires about 500 seconds, on average, to reach a gap under 0.1%. It is very encouraging, however, to witness this not-so-bad behavior on instances that are different (no item profits) from those the algorithms were developed for. This, combined to the excellent performance on the other instance classes, shows the efficiency of the proposed algorithms and the interest to continue the research in this area.

7 Conclusion

We introduced the Generalized Bin Packing Problem, a new packing problem where, given a set of items characterized by volume and profit and a set of bins with given volume and cost, one aims to select the subsets of profitable items and appropriate bins to optimize an objective function obtained by combining the cost of using the bins and the profit derived by loading the selected items. The GBPP generalizes many other packing problems, including Bin Packing and Variable Cost and Size Bin Packing, as well as Multiple Homogeneous and Heterogeneous Knapsack, and is relevant for many transportation and logistics planning problems.

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 more interesting in algorithmic terms, conducting 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 also introduced a large number of new instances. The analysis of the results of extensive computational experiments showed that the proposed procedures are quite efficient and the bounds are tight.

Acknowledgments

While working on this project, the second author was the NSERC Industrial Research Chair on 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 supported by the Ministero dell'Istruzione, Università e Ricerca (MIUR) (Italian Ministry of University and Research), under the Progetto di Ricerca di Interesse Nazionale (PRIN) 2007, "Optimization of Distribution Logistics", 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

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.

A. Atamturk and M. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140:67–124, 2005.

J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52 (8):928–936, 2001.

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

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: 1295–1303, 2007a.

T. G. Crainic, G. Perboli, M. Pezzuto, and R. Tadei. New bin packing fast lower bounds. *Computers and Operations Research*, 34:3439–3457, 2007b.

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.

F. de la Vega and W. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.

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

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

N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proc. 23rd Annual Symp. Found. Comp. Sci. (FOCS 1982)*, pages 312–320, 1982.

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

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

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403–2435, 2007.

D. Pisinger and M. Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2:154–167, 2005.

P. Schwerin and G. Wäscher. The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operations Research*, 4:377–389, 2006.

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.

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

G. Wäscher, H. Haussner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183:1109–1130, 2007.