CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

# Efficient Heuristics for the Variable Size Bin Packing Problem with Fixed Costs

**Teodor Gabriel Crainic**
**Guido Perboli**
**Walter Rei**
**Roberto Tadei**

**March 2010**

**CIRRELT-2010-18**

**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

# Efficient Heuristics for the Variable Size Bin Packing Problem with Fixed Costs

**Teodor Gabriel Crainic[1,2,*], Guido Perboli[1,3], Walter Rei[1,2], Roberto Tadei[3]**

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

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

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

**Abstract.** We consider the Variable Size Bin Packing Problem with Fixed Costs, a generalization of the well-known bin packing problem where a set of items must be packed within a set of heterogeneous bins characterized by possibly different volumes and fixed costs. The objective of the problem is to select the bins to pack all items while minimizing the total cost incurred for the selected bins. We present new heuristic solution methods for the problem that integrate lower and upper bound techniques. Extensive numerical tests conducted on instances with up to 1000 items show the effectiveness of these methods in terms of computational effort and solution quality.

**Keywords**. Variable-size bin packing with fixed costs, heuristics, lower bounds, upper bounds.

**Acknowledgements.** We express our gratitude to Dr. Correia who graciously provided us with the sets of test instances. Funding for this project has been provided by the Politecnico di Torino 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.



---

* Corresponding author: Teodor.Gabriel@cirrelt.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2010

# 1 Introduction

Bin packing problems aim to load a series of items into a number of bins, such that the packing of each bin is feasible with respect to a number of restrictions, e.g., capacity limits or balancing constraints, while optimizing a given objective function, e.g., minimizing the total number of used bins [5, 3, 11]. Yet, despite their interest as combinatorial optimization problems and their importance to a wide gamut of applications, e.g., planning of telecommunication, transportation, production, and logistics/supply-chain systems, not all packing problem classes have received adequate attention.

We aim to contribute to address this issue by focusing on the one dimensional *Variable Size Bin Packing Problem with Fixed Cost* (VSBPPFC). In this setting, a finite set of items must be packed within a finite set of heterogeneous bins, characterized by possibly different volumes (capacity) and fixed costs. The objective is to select the bins to pack all items while minimizing the total cost incurred for the selected bins. Most solution methods in the literature rely either on decomposition techniques [6, 8, 1] or on reformulations solved using commercial MIP software [2] and require significant computation times when applied to large instances. This computational effort makes them difficult to use in actual planning applications, where efficient, i.e., *fast* and *accurate*, solution methods are crucial to address the VSBPPFC subproblems that must be solved repeatedly.

We propose new heuristic solution methods for the VSBPPFC that successfully tackle the challenge of efficiency by using lower and upper bound techniques to select bins and pack items. We test these heuristics on a large set of instances, and compare our results to those of the best methods in the literature [2, 6]. This numerical analysis indicates that the proposed heuristics find very good solutions extremely fast, even when addressing instances with up to 1000 items.

The paper is organized as follows. Section 2 presents the VSBPPFC model and briefly reviews the methods proposed in the literature to address it. Lower bounds and the corresponding heuristics are introduced in Sections 3 and 4, respectively. Section 5 is dedicated to the presentation and analysis of the computational results. We conclude in Section 6.

# 2 The VSBPPFC Model and Related Work

Let $\mathcal{I}$ ($|\mathcal{I}| < \infty$) be the set of items to be loaded. Each item $i \in \mathcal{I}$ has a volume $v_i$. Let $\mathcal{J}$ ($|\mathcal{J}| < \infty$) be the set of available bins and let $V_j$ and $c_j$ be the volume and cost of bin $j \in \mathcal{J}$, respectively. Without any loss of generality, let us assume that the volumes and

costs associated with the bins and items are integers.

Define the *bin-selection* variables $y = (y_1, y_2, \ldots, y_{|\mathcal{J}|})$, where $y_j = 1$, if bin $j$ is selected and $y_j = 0$, otherwise, and the *item-to-bin assignment* variables $x_{ij}$, $\forall i \in \mathcal{I}$, $\forall j \in \mathcal{J}$, where $x_{ij} = 1$, if item $i$ is loaded into bin $j$ and $x_{ij} = 0$, otherwise. Let $z(y) = \sum_{j \in \mathcal{J}} c_j y_j$. Then, the VSBPPFC model can be formulated as:

$$\begin{aligned}
\min \quad & z(y) && && (1)\\
\text{s.t.} \quad & \textstyle\sum_{j \in \mathcal{J}} x_{ij} = 1, && \forall i \in \mathcal{I}, && (2)\\
& \textstyle\sum_{i \in \mathcal{I}} v_i x_{ij} \leq V_j y_j, && \forall j \in \mathcal{J}, && (3)\\
& x_{ij} \in \{0,1\}, && \forall i \in \mathcal{I}, \quad \forall j \in \mathcal{J}, && (4)\\
& y_j \in \{0,1\}, && \forall j \in \mathcal{J}. && (5)
\end{aligned}$$

The objective function (1) minimizes the total fixed cost of the selected bins. Constraints (2) ensure that each item $i \in \mathcal{I}$ is packed within exactly one bin, while constraints (3) make sure that the total volume of the items packed into bin $j \in \mathcal{J}$ does not exceed its volume $V_j$. Relations (4) and (5) enforce the integrality requirements for all decision variables.

A somewhat small number of studies have addressed the VSBPPFC. In his Ph.D. thesis dedicated to packing and scheduling problems, Monaci presents lower bounds and heuristic and exact solution methods for the VSBPPFC with bin fixed costs equal to the volumes of the bins [6] (neither the results, nor the instance set have been published). Seiden *et al.* [9] consider the on-line version of the problem and propose upper and lower bounds. Kang and Park [4] develop two greedy algorithms for the special case of the VSBPPFC, where the cost of a unit of bin volume does not increase as the bin volume increases, and analyze their performance on instances with divisibility constraints (on both item and bin sizes, on only the bin sizes, and in the general case where no divisibility constraints are present). An integer-linear formulation for the two-dimensional VSBPPFC is proposed by Pisinger and Sigurd [8], together with lower bounds based on Dantzig-Wolfe decomposition and an exact branch-and-price algorithm. Alves and Valério de Carvalho [1] propose a series of strategies aimed at accelerating the column generation approach for the same problem. Finally, Correia *et al.* [2] discretize the VSBPPFC formulation, and propose valid inequalities to improve the quality of the lower bounds obtained from the linear relaxation of the resulting model. The authors analyze the quality of the lower bounds on a large set of instances with up to 1000 items.

# 3 Lower bounds for the VSBPPFC

We present a series of lower bounds for the VSBPPFC that provide the means to measure the solution quality of the various procedures and are also used as building blocks for the heuristics proposed in Section 4.

Let $y^\star = (y_1^\star, y_2^\star, \ldots, y_{|\mathcal{J}|}^\star)$ stand for an optimal selection of bins yielded by the formulation (1)-(5), with $z^\star$ the associated optimal value of the objective function. Consider the following problem, originally proposed in [6] for the special case of the VSBPPFC with bin fixed costs equal to be volumes of the bins:

$$\begin{aligned} \min \quad & z(\hat{y}) && (6)\\ \text{s.t.} \quad & \textstyle\sum_{j\in\mathcal{J}} V_j \hat{y}_j \geq \sum_{i\in\mathcal{I}} v_i, && (7)\\ & \hat{y}_j \in \{0,1\}, \qquad \forall j \in J. && (8) \end{aligned}$$

Let $\hat{y}^\star = (\hat{y}_1^\star, \hat{y}_2^\star, \ldots, \hat{y}_{|J|}^\star)$ represent an optimal solution to problem (6)-(7). The bin selection $\hat{y}^\star$ minimizes the total fixed cost and ensures that the total volume of the selected bins is at least as large as the total volume of the items. The single item-to-bin requirement (2) is relaxed, however, and thus $\hat{y}^\star$ may not be feasible for the original problem. Obviously, $z(\hat{y}^\star) \leq z^\star$.

Performing the substitution

$$\hat{y}_j = 1 - u_j, \tag{9}$$

within problem (6)-(8), one obtains

$$\begin{aligned} \max \quad & z(u) && (10)\\ \text{s.t.} \quad & \textstyle\sum_{j\in\mathcal{J}} V_j u_j \leq \sum_{j\in\mathcal{J}} V_j - \sum_{i\in\mathcal{I}} v_i, && (11)\\ & u_j \in \{0,1\}, \forall j \in \mathcal{J}. && (12) \end{aligned}$$

Formulation (10)-(12) corresponds to a 0/1 knapsack problem. Therefore, by applying (9), one can derive an optimal solution to (6)-(8) by solving the knapsack problem (10)-(12) using any available exact method [5]. Furthermore, any lower bound $LB(z(u))$ obtained for problem (10)-(12) can also be used to produce a valid lower bound $LB(z(\hat{y}))$ for problem (6)-(8): $LB(z(\hat{y})) = \sum_{j\in\mathcal{J}} c_j - LB(z(u))$. We denote $LB_1$ the lower bound obtained from the original model (6)-(8).

The lower bound provided by (6)-(8) can be improved by considering that bins may be grouped according to the distinct values of their volumes [6]. Let $\mathcal{K}$ define the set of distinct bin types relative to the bin volumes, i.e., $\forall k_1, k_2 \in \mathcal{K}$, $V_{k_1} \neq V_{k_2}$. Let $M_k,\ k \in \mathcal{K}$ be the maximum ("best") filling ratio for the bin type $k$ given the set of

items $\mathcal{I}$, obtained by solving the knapsack problem $M_k = \max\Big\{ \sum_{i\in\mathcal{I}} v_i x_i \mid \sum_{i\in\mathcal{I}} v_i x_i \leq V_k,\ x_i \in \{0,1\}, \forall i \in \mathcal{I}\Big\}$. Clearly, $M_k \leq V_k$, $\forall k \in \mathcal{K}$. Therefore, by replacing the bin volumes by their associated maximum filling ratios in (10)-(12), one obtains an improved lower bound on (1)-(5) by solving

$$\min \quad z(\tilde{y}) \tag{13}$$
$$\text{s.t.} \quad \sum_{j\in\mathcal{J}} M_j \tilde{y}_j \geq \sum_{i\in\mathcal{I}} v_i, \tag{14}$$
$$\tilde{y}_j \in \{0,1\}, \forall j \in J. \tag{15}$$

Once again, substitution (9) transforms the formulation (13)-(15) into a knapsack problem, which may be used to compute lower bounds. We denote $LB_2$ the lower bound obtained from the formulation (13)-(15).

$LB_1$ can be further improved by considering that each item must be compulsory loaded. Let us define, for each item $i$ and each bin type $k$, $M_{ik} = \max\Big\{ \sum_{j\in\mathcal{I}\setminus i} v_i x_i \mid \sum_{i\in\mathcal{I}} v_i x_i \leq V_k - v_i,\ x_i \in \{0,1\}, \forall i \in \mathcal{I}\setminus i\Big\}$. Each $M_{ik}$ gives the maximum filling of a bin of type $k$ when the item $i$ is loaded into it. Thus, $t_i = \min_{k\in\mathcal{K}}\{V_k - M_{ik}\} - v_i$ represents the loss of volume due to the compulsory loading of item $i$ and a new lower bound can be computed by applying $LB_1$ to the instance where the item volumes are set to $\tilde{v}_i = v_i + t_i$.

The task of computing the $M_{ik}$ indexes can be computationally heavy, however, requiring solving a knapsack for each item and bin type. We therefore use an approximation

$$\overline{M(p)}_{ik} = \begin{cases} M_{ik} & if\ v_i + \sum_{j=1}^{p} v_j > V_k \ \text{ and } \ v_i + \sum_{j=1}^{p-1} v_j \leq V_k, \\ V_k - v_i & \text{otherwise,} \end{cases} \tag{16}$$

where items are ordered by the non-decreasing values of their volumes $v_i$. The idea is that,in the best case, one cannot load more than $p$ items including a large item $i$ into a bin of type $k$ when 1) loading the smallest $p$ items together with item $i$ overloads the bin, but 2) loading only the $p-1$ smallest items together with item $i$ does not. Computing $M_{ik}$ in this case is then easy, since one has only to evaluate all the $p-tuples$ containing the item $i$, which is $O(n^{p-1})$. Otherwise, we assume item $i$ to be "small" and set the associate loss of space to null ($t_i = 0$).

The use of $\overline{M(p)}_{ik}$ is profitable when the value of $p$ is small, e.g., $p = 2$ for the problems we tested (see Section 5). We refer to $LB_3$, the lower bound obtained by applying $LB_1$ to the instances where the item volumes are set to $\tilde{v}_i = v_i + t_i$ and $t_i$ is defined according to $\overline{M(2)}_{ik}$ (notice that for any solution method, these indexes need to be computed only once, during initialization, which makes $LB_3$ suitable even for Branch & Bound algorithms).

# 4 Heuristics for the VSBPPFC

In this section, we present new heuristics for the VSBPPFC. The heuristics combine information yielded by the lower bound computations and extensions of a number of fundamental concepts in the bin packing and knapsack methodology.

We thus adapt the well-known *Best First Decreasing* (BDF) loading heuristic, which offers good performance for the classical bin packing problem, and loads each item into the "best" bin, i.e., the bin with the maximum free space (defined as the bin volume minus the sum of the volumes of the items loaded into it) once the item is loaded [5]. The BFD we propose for the VSBPPFC, denoted the *Adapted BFD* ($A-BFD$) heuristic, first sorts the items according to the non-increasing order of their volumes and, then, sequentially loads them (Algorithm 1 displays the pseudocode). For each item, one first attempts to load it into the "best" already-selected bin, that is, the bin maximizing the *merit function* computing the free bin space as defined above. If the item cannot be loaded into an already-selected bin, a new bin is selected and the item is loaded into it.

An issue particular to the VSBPPFC, but irrelevant for the classical bin packing problem where bins are homogeneous, is how to choose a new bin, when required. Inspired by the item-selection rule for knapsack problems, we select bins according to the non-increasing order of their unit cost / volume ratios, $c_j/V_j$, and in the non-decreasing order of their volumes when the unit costs are equal.

Considering the bins according to this order generally yields good solutions, but may falter when the last items are considered. Indeed, when a new bin needs to be selected for an item toward the end of the list, the selected bin might have a volume much larger than that of the item, even though its cost/volume ratio is good. Moreover, few items might be left to take advantage of this volume. A bin with a worst ratio but an absolute smaller cost, might be appropriate in this case, and we implement a post-processing procedure that attempts to improve the solution by evaluating such possible bin swaps. The procedure iteratively examines each selected bin $j \in \mathcal{J}$ with its cost $c_j$ and loaded volume $U_j$ defined as the sum of the volumes of the items assigned to it. Then, if it exists an unused bin $k \in \mathcal{J},\ k \neq j$ such that $V_k \geq U_j$ and $c_k < c_j$, it transfers the items from bin $j$ to bin $k$ and discards the former.

Notice that any of the three lower bounds presented in Section 3 yields a set of bins with total cost equal to the lower bound. We may use this information to initialize the set of bins used in the heuristic and thus obtain a variant of $A-BFD$. More precisely, the *LB-Based BFD* ($LB-BFD$) heuristic works as follows:

- Consider the set of the bins given by the selected lower bound, generically denoted $LB$;

**Algorithm 1** $A-BFD$

**Input** $\mathcal{I}$ : Items to be accommodated into the bins
**Input** $\mathcal{K}$ : Bin groups (types) available to load the items

$\mathcal{S}$ : Set of selected bins (empty at the beginning)

Sort the items in $\mathcal{I}$ according to non-increasing order of their volume
Sort the bins in $\mathcal{K}$ according to non-increasing order of the ratio $c_j/V_j$, and non-decreasing order of $V_j$ when the unit costs $c_j$ are equal)
$\mathcal{S} = \{\emptyset\}$
**for all** $i \in \mathcal{I}$ **do**
  **if** $i$ can be accommodated into a bin in $\mathcal{S}$ **then**
    Accommodate $i$ into the best bin $b \in \mathcal{S}$
  **else**
    $\mathcal{S} = \mathcal{S} \cup \{b'\}$, where $b'$ is the first bin in the ordered list $\mathcal{K}$
    Accommodate $i$ into $b'$
  **end if**
**end for**
**for all** $j \in \mathcal{S}$ **do**
  **for all** $k \in \mathcal{K} \setminus \mathcal{S}$ **do**
    $U_j = \sum_{i \text{ loaded in } j} v_i$
    **if** $V_k \geq U_j$ 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**

**return** $\mathcal{S}$

- Initialize the solution with a percentage $p \in (0, 1]$ of these bins (bins are added empty);
- Order the remaining bins as described above and apply $A-BFD$ to build a feasible solution.

A second variant of $A - BFD$ also starts from the optimal solution associated to a lower bond $LB$ in order to choose the bins in the solution, but recomputes the bound when items cannot be loaded into already-selected bins. The heuristic, denoted $ITER-BFD$, iteratively adds a subset of the bins given by $LB$. Then, the new bins are loaded by considering the items according to their non-increasing order of volumes. When an item cannot be loaded, a partial instance is built out of the items not yet loaded and the bins still unused, and a new bound $LB$ is computed. The procedure stops when the item list is empty or, if after a maximum number of iterations $\overline{k}$ some items are still unloaded, by applying the $A - BFD$ heuristic.

# 5 Computational results

In this section we present and analyze the results of the computational experiments. The first objective is to asses the relative performance of the proposed lower and upper bounds. The second is to evaluate the efficiency of our procedures by comparing our best results to the best methods proposed in the literature.

The new lower and upper bounds are implemented in C++. Experiments were performed on a Pentium IV 3GhZ workstation. The knapsack instances generated by the $LB_1$ and $LB_2$ bounds were solved to optimality by means of the algorithm presented in [7].

The following sets of problem instances were used in the experiments:

**Set1.** Instances from [2]. Five instances were randomly generated for each combination of the following parameters:

- Number of items in the set $\{100, 200, 500, 1000\}$.
- Item volumes randomly generated according to a (discrete) uniform distribution in the set $\{1, 2, ..., 20\}$.
- Five bin capacity cases: 1) all bins with the same capacity of 150; 2) three different capacities, 100, 200, and 300; 3) six different capacities, 50 to 300 by increment of 50; 4) twelve different capacities, from 25 to 300 by increment of 25; 5) all bins with different capacity from 60 to 330 by increment of 5.
- Bin fixed cost set to $100\sqrt{V_j}$ for each bin $j$.
- For the number of bins for each type, the minimum number $D$ required to load all the items is computed and, then, instances with $D$ and $D+1$ bins are built.

**Set2.** Instances proposed in [6] and regenerated for [2], derived from classical bin packing instances for the item-volume distribution and relations to the bin volume [10]. Ten instances were randomly generated for each combination of the following parameters:

- Item volume: three types with volumes uniformly distributed in [1; 100], [20; 100], and [50; 100], respectively.
- Number of bin types: 3 (capacities equal to 100, 120 and 150, respectively) and 5 (capacities equal to 60, 80, 100, 120 and 150, respectively).
- Number of items: 25, 50, 100, 200, and 500.

We first examine the relative performance of the different versions of the lower and upper bound procedures. With respect to the latter, we report results for the $LB-BFD$

and $ITER-BFD$ variants using $LB_1$ only, because there was no difference in our final results when one of the two other lower bounds was used. Moreover, we report the comparison results on instances of Set 2 only, because both $LB1$ and the $A-BFD$ heuristic solved to optimality all the instances of Set 1. We further discuss this observation in the second part of the section.

| Type/$p$ | 0.10 | 0.15 | 0.20 | 0.25 | 0.30 | 0.35 | 0.40 | 0.45 | 0.50 | 0.55 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.2 | 1.17 | 1.3 | 1.42 | 1.45 | 1.39 | 1.5 | 1.63 | 1.81 | 1.85 |
| 2 | 2.5 | 2.57 | 2.61 | 2.61 | 2.61 | 2.68 | 2.83 | 3.04 | 3.33 | 3.94 |
| 3 | 1 | 0.97 | 1.07 | 1.24 | 1.43 | 1.55 | 1.64 | 2.03 | 2.66 | 3.12 |
| Mean | 1.6 | 1.57 | 1.66 | 1.76 | 1.83 | 1.87 | 1.99 | 2.23 | 2.6 | 2.97 |

| Type/$p$ | 0.60 | 0.65 | 0.70 | 0.75 | 0.80 | 0.85 | 0.90 | 0.95 | 1.00 | Best |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2.1 | 2.39 | 2.52 | 2.52 | 2.73 | 2.76 | 2.56 | 2.8 | 2.7 | 0.64 |
| 2 | 4.2 | 4.44 | 4.39 | 4.53 | 4.51 | 4.54 | 4.61 | 4.93 | 5.03 | 1.65 |
| 3 | 3.7 | 4.17 | 4.5 | 4.93 | 5.44 | 5.88 | 6.27 | 6.66 | 6.91 | 0.21 |
| Mean | 3.3 | 3.67 | 3.81 | 3.99 | 4.23 | 4.4 | 4.48 | 4.8 | 4.88 | 0.83 |

Table 1: Set 2: Mean optimality gaps (%) for the $LB-BFD$ heuristic

Table 1 reports the optimality-gap performance of the $LB-BFD$ heuristic while varying $p$, the percentage of bins from the optimal solution of $LB_1$ used to initialize the heuristic solution, from 10% to 100%. The first column displays the item volume type, while Columns 2 to 19 report the mean percentage deviation of $LB-BFD$ from the optimal solutions for the various values of $p$. The best mean values appear to be obtained for $10\% \leq p \leq 40\%$, but no setting of this parameter seems to dominate the others. Computing $A-BFD$ (or $LB-BFD$) requires less than 0.01 seconds CPU for the largest instances, however. In fact, computing $LB_1$ together with all the solution obtained by $LB-BFD$ for varying $p$ requires less that $10^{-2}$ in the worst case (most of it required to compute $LB_1$). We therefore computed a composite solution as the best solution among those with $10\% \leq p \leq 40\%$ and we report this value in the last column. This composite implementation of the heuristic yields solutions that are less than 1% from the optimal solution.

| Type/$\overline{k}$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Best |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2.28 | 1.75 | 1.80 | 1.85 | 1.89 | 1.81 | 1.68 | 1.78 | 1.64 | 0.92 |
| 2 | 3.70 | 3.22 | 3.20 | 3.17 | 3.16 | 3.06 | 2.90 | 2.89 | 2.87 | 2.17 |
| 3 | 5.57 | 5.19 | 4.60 | 4.09 | 3.65 | 3.48 | 3.50 | 3.41 | 3.26 | 2.68 |
| Mean | 3.85 | 3.38 | 3.20 | 3.04 | 2.90 | 2.78 | 2.69 | 2.69 | 2.59 | 1.93 |

Table 2: Set 2: Mean optimality gaps (%) for the $ITER-BFD$ heuristic

Table 2 reports the mean optimality gaps obtained by the $ITER-BFD$ heuristic while varying the maximum number of iterations $\overline{k}$ (Columns 2 to 10), as well as the composite solution (Column 12) built on the same idea as previously. The results indicate that $ITER-BFD$ has a worst behavior than $LB-BFD$, with an average optimality gap of about 2%. The computational time, although small (less than 0.1 seconds for the largest instances) is also larger than for $LB-BFD$, due to the necessity to compute a knapsack problem at the end of each iteration.

Table 3 sums up the performance results of the lower and upper bound procedures introduced in this paper, together with those of $Comp_BFD$, a composite heuristic selecting the best solution among those of the three upper bound procedures. Columns 1 and 2 present the volume type and the number of items, respectively, while the following columns display the mean optimality gaps for the seven procedures. The best settings, as reported in Tables 1 and 2, were used for $LB-BFD$ and $ITER-BFD$, respectively.

The lower bound procedures are performing very well for instances of type 1 and 2, while the gap increases slightly for instances of type 3. The latter are characterized by a peculiar item-to-bin volume relation, however. Indeed, the minimum item volume is 50 and, thus, at most three items can be loaded into each bin type. The loading patterns for each bin type are therefore polynomially limited, facilitating column generation-based methods.

With respect to the upper bounds, the best mean results were obtained by the $A-BFD$ and $LB-BFD$ heuristics. The most critical instances from the upper bound point of view are those of type 2, characterized by the presence of medium-sized items (volumes generated in the interval $20-100$). But, as often in numerical experiments, the means do not tell the whole story. Thus, just considering the mean, $ITER-BFD$ is outperformed by the other two heuristics. An instance-by-instance verification of the results shows, however, that $ITER-BFD$ performs better on the instances for which $A-BFD$ and $LB-BFD$ yield their worst results. The upper bound procedures thus appear "complementary" and, because their respective computational times are small, we propose to compute the three values and select the best. The $Comp_BFD$ computes this best result, and yields a mean optimality gap of 0.78% with a computation effort of less than 0.1 seconds in the worst case.

In the second part of this analysis, we present a comparison between our bounds and best results from the literature [6, 2]. The optimal solutions and the lower bound values obtained by Correia et al. are taken from the literature [2], while the column generation of Monaci [6] has been reimplemented in order to have a better insight about computational times and results (the detailed results are not available).

The comparisons on the instances from Set 1 are not reported, because $LB_1 = A-BFD$ for all instances, i.e., all instances are solved to the optimum by using the proposed

| Type | n | $LB_1$ | $LB_2$ | $LB_3$ | $A-BFD$ | $LB-BFD$ | $ITER-BFD$ | $Comp_BFD$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 25 | 0.21 | 0.21 | 0.21 | 1.41 | 1.41 | 1.88 | 1.28 |
| | 50 | 0.02 | 0.02 | 0.02 | 0.95 | 0.95 | 1.50 | 0.93 |
| | 100 | 0.00 | 0.00 | 0.00 | 0.49 | 0.49 | 0.60 | 0.45 |
| | 200 | 0.00 | 0.00 | 0.00 | 0.32 | 0.27 | 0.39 | 0.25 |
| | 500 | 0.00 | 0.00 | 0.00 | 0.10 | 0.10 | 0.25 | 0.09 |
| Mean | | 0.05 | 0.05 | 0.05 | 0.66 | 0.64 | 0.92 | 0.60 |
| 2 | 25 | 1.14 | 1.08 | 1.08 | 2.13 | 1.92 | 2.66 | 1.64 |
| | 50 | 0.29 | 0.29 | 0.29 | 1.93 | 1.81 | 2.45 | 1.76 |
| | 100 | 0.05 | 0.05 | 0.05 | 1.84 | 1.78 | 1.79 | 1.59 |
| | 200 | 0.00 | 0.00 | 0.00 | 1.51 | 1.51 | 1.94 | 1.43 |
| | 500 | 0.00 | 0.00 | 0.00 | 1.47 | 1.26 | 2.03 | 1.24 |
| Mean | | 0.30 | 0.28 | 0.28 | 1.78 | 1.65 | 2.17 | 1.53 |
| 3 | 25 | 3.08 | 1.81 | 1.65 | 0.14 | 0.14 | 1.10 | 0.14 |
| | 50 | 1.86 | 1.43 | 1.39 | 0.28 | 0.28 | 1.54 | 0.27 |
| | 100 | 1.57 | 1.44 | 1.44 | 0.20 | 0.20 | 3.05 | 0.20 |
| | 200 | 0.87 | 0.86 | 0.86 | 0.33 | 0.33 | 3.72 | 0.33 |
| | 500 | 0.85 | 0.85 | 0.85 | 0.09 | 0.09 | 3.99 | 0.09 |
| Mean | | 1.64 | 1.28 | 1.23 | 0.21 | 0.21 | 2.68 | 0.2 |
| Overall mean | | 0.66 | 0.54 | 0.52 | 0.88 | 0.83 | 1.93 | 0.78 |

Table 3: Set 2: Average optimality gaps (%) for the upper and lower bound procedures

heuristic. We report in Table 4, however, the results of the instances of Set 1 not solved to optimality in [2]. For each instance, we give in Columns 1-4, the instance name, the number of bin types, the number of bins per type, and the number of items, respectively. The best-known feasible solution reported in the literature is displayed in Column 5, while the values of $LB_1$ and $A-BFD$ are reported in the last two columns. The results show that we obtained or improved (instance *pbin_1000_2*) all the best-known solutions, which are now proved to be optimal. Notice that the column generation by Monaci yielded a mean optimality gap of 0.23% on the instances of this set [6]. The proposed lower and upper bound procedures are thus performing very well on the instances introduced in [2], a conclusion even more impressive when one considers that this performance is obtained in less than 0.01 CPU seconds in the worst case.

| Instance | m | D | n | Best Known | $LB_1$ | $A-BFD$ |
|---|---|---|---|---|---|---|
| *pbin_500_4* | 6 | 3 | 500 | 35313 | 35313 | 35313 |
| *pbin_500_4* | 6 | 3 | 500 | 35313 | 35313 | 35313 |
| *pbin_500_5* | 6 | 3 | 500 | 34089 | 34089 | 34089 |
| *pbin_1000_1* | 6 | 12 | 1000 | 68740 | 68740 | 68740 |
| *pbin_1000_2* | 6 | 7 | 1000 | 69345 | **68828** | **68828** |
| *pbin_1000_3* | 6 | 12 | 1000 | 69964 | 69964 | 69964 |
| *pbin_1000_4* | 6 | 7 | 1000 | 69121 | 69121 | 69121 |
| *pbin_1000_1* | 12 | 6 | 1000 | 72001 | 72001 | 72001 |
| *pbin_1000_2* | 12 | 6 | 1000 | 72160 | 72160 | 72160 |
| *pbin_1000_3* | 12 | 6 | 1000 | 73160 | 73160 | 73160 |

Table 4: Set 1: New optimal solutions

Table 5 summarizes the gaps (in %) of the lower bounds from the optimal solutions of instances in Set 2. The first and the second column report the volume-generation type and the number of items, respectively. The following columns report the mean gaps over ten instances obtained by the lower bound $LB_M$ by Monaci [6], $LB_C$ by Correia et al. [2], and the lower bounds we propose, $LB_1$, $LB_2$, and $LB_3$. The last row of the a type reports the mean deviations from the optimal values for all the instances of the set.

The best performance is offered by $LB_C$, but the application of this procedure is limited by the size of the MIP models involved. Thus, for example, it could not address instances with 500 items. With respect to the other bounds, one observes that $LB_1$, $LB_2$, and $LB_3$ performed better than $L_M$ on instances of type 1, characterized by item volumes in a broad interval ([1; 100]), and on the larger instances of type 2. As discussed earlier on, the particular item-to-bin volume relation characterizing the other instances, favors column generation-based procedures and heavily penalizes the proposed lower bounds.

Turning to computational times, $LB_C$ requires 300 CPU seconds on average and up to 3000 CPU seconds for their largest instances (200 items; instances with 500 items

were not considered by the authors) using a 2.4 GhZ Pentium IV workstation. Moreover, the computational effort of $LB_C$ seems to increase when the packing component of the problem becomes the main part as opposed to the bin-selection part. On the other hand, the column generation method of [6] is quite effective. Thus, an average of 2 CPU seconds were required for instances with 500 items, for which an average of 45 columns were generated and, thus, 45 knapsack problems (plus the corresponding LPs) were solved. Our lower bound procedures are even more effective, however, requiring to solve only one knapsack instance for $LB_1$ and at most 6 instances for $LB_2$, with a worst case of 0.01 seconds.

Consequently, $LB_1$ achieves a performance of over 99%, compared to the other procedures, in a negligible computational time. This identifies it as a good candidate to rapidly obtain very good solutions and for use in more complex problem-solving settings, e.g., simulations and optimization frameworks where the VSBPPFC appears as a subproblem in meta-heuristics or Branch & Bound schemes.

| Type | n | $LB_c$ | $LB_m$ | $LB_1$ | $LB_2$ | $LB_3$ |
|---|---|---|---|---|---|---|
| 1 | 25 | 0,00 | 0,38 | 0,21 | 0,21 | 0,21 |
| | 50 | 0,00 | 0,19 | 0,02 | 0,02 | 0,02 |
| | 100 | 0,00 | 0,08 | 0,00 | 0,00 | 0,00 |
| | 200 | 0,00 | 0,05 | 0,00 | 0,00 | 0,00 |
| | 500 | N/A | 0,02 | 0,00 | 0,00 | 0,00 |
| Average | | 0,00 | 0,14 | 0,05 | 0,05 | 0,05 |
| 2 | 25 | 0,00 | 0,22 | 1,14 | 1,08 | 1,08 |
| | 50 | 0,00 | 0,14 | 0,29 | 0,29 | 0,29 |
| | 100 | 0,00 | 0,07 | 0,05 | 0,05 | 0,05 |
| | 200 | 0,00 | 0,03 | 0,00 | 0,00 | 0,00 |
| | 500 | N/A | 0,02 | 0,00 | 0,00 | 0,00 |
| Average | | 0,00 | 0,09 | 0,30 | 0,28 | 0,28 |
| 3 | 25 | 0,00 | 0,09 | 3,08 | 1,81 | 1,65 |
| | 50 | 0,00 | 0,10 | 1,86 | 1,43 | 1,39 |
| | 100 | 0,00 | 0,05 | 1,57 | 1,44 | 1,44 |
| | 200 | 0,00 | 0,03 | 0,87 | 0,86 | 0,86 |
| | 500 | N/A | 5,02 | 0,85 | 0,85 | 0,85 |
| Average | | 0,00 | 1,06 | 1,64 | 1,28 | 1,23 |

Table 5: Set 2: Lower bound optimality gap (%) comparison

# 6 Conclusion

We introduced very efficient upper and lower bounds for the VSBPPFC, a problem combining bin packing and knapsack characteristics, with applications in many important areas. The bounds present an accuracy of more than 99% with a computational effort several order of magnitude lower compared to state-of-the-art methods.

# Acknowledgments

We express our gratitude to Dr. Correia who graciously provided us with the sets of test instances.

While working on this project, the first 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. Funding for this project has been provided by the Politecnico di Torino 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] Alves, C., Valérion de Carvalho, J.M.: Accelerating column generation for variable sized bin-packing problems. European Journal of Operational Research **183**, 1333–1352 (2007)

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

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

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

[5] Martello, S., Toth, P.: Knapsack Problems - Algorithms and computer implementations. John Wiley & Sons, Chichester, UK (1990)

[6] Monaci, M.: Algorithms for packing and scheduling problems. Ph.D. thesis, Universit di Bologna (2002)

[7] Pisinger, D.: A minimal algorithm for the 0-1 knapsack problem. Operations Research **45**, 758–767 (1997)

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

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

[10] Vanderbeck, F.: Computational study of a column generation algorithm for bin packing and cutting stock problems. Mathematical Programming pp. 565–594 (1999)

[11] Washer, G., HauBner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research **183**, 1109–1130 (2007)