Efficient lower bounds and heuristics for the variable cost and size bin packing problem

Author's Accepted Manuscript

www.elsevier.com/locate/cor

Efficient lower bounds and heuristics for the variable cost and size bin packing problem

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

PII: S0305-0548(11)00004-9
DOI: doi:10.1016/j.cor.2011.01.001
Reference: CAOR 2719

To appear in: *Computers & Operations Research*

Received date: 12 October 2010
Revised date: 17 December 2010
Accepted date: 4 January 2011

Cite this article as: Teodor Gabriel Crainic, Guido Perboli, Walter Rei and Roberto Tadei, Efficient lower bounds and heuristics for the variable cost and size bin packing problem, *Computers & Operations Research*, doi:10.1016/j.cor.2011.01.001

Efficient Lower Bounds and Heuristics for the Variable Cost and Size Bin Packing Problem

Teodor Gabriel Crainic[c,1,*], Guido Perboli[a,1], Walter Rei[c,1], Roberto Tadei[a]

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

Abstract

We consider the Variable Cost and Size Bin Packing Problem, a generalization of the well-known bin packing problem, where a set of items must be packed into a set of heterogeneous bins characterized by possibly different volumes and fixed selection costs. The objective of the problem is to select bins to pack all items at minimum total bin-selection cost. The paper introduces lower bounds and heuristics for the problem, the latter integrating 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. We also provide a systematic numerical analysis of the impact on solution quality of the bin selection costs and the correlations between these and the bin volumes. The results show that these correlations matter and that solution methods that are unbiased toward particular correlation values perform better.

Keywords: Variable cost and size bin packing, Heuristics, Lower Bounds, Upper Bounds

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 load balancing constraints, while optimizing a given objective, such as the minimization of the total number of used bins [1, 2, 3].

Bin packing problems belong to the core of classical combinatorial optimization problems. They are also of great practical interest to, e.g., planning of telecommunication, transportation, production, and logistics/supply-chain systems. Although a great body of work has been conducted on the topic [3], most developments ignored economic considerations such as bin or item costs, focusing instead on the physical characteristics associated with such problems, e.g., packing constraints on multiple dimensions..

We aim to contribute to address this issue by focusing on a problem setting including both physical and economic attributes. More precisely, we address bin packing problems where a finite set of items must be packed into a finite set of heterogeneous bins, characterized by possibly different volumes (capacity) and selection fixed costs. The objective is to select the bins to pack all items at minimum total bin selection cost.

The previous contributions to this problem setting were referred to in the literature as the *Variable Size Bin Packing Problem* (VSBPP) [3].. It should be noted, however, that an important assumption was made in this literature regarding the values of the bin fixed costs, which were defined as equal to the volumes of the corresponding bins or, at best, as directly proportional to them. This is a serious limitation because, in practical applications, other than its volume, the cost of a bin is influenced by many other external factors, as discussed in Section 2.

Our objective is to lift this hypothesis and address the problem in all generality by considering that fixed costs are attributes associated with available bins that may, or may not, be correlated to their volumes. We refer to this version

*Corresponding author
Email addresses: TeodorGabriel.Crainic@cirrelt.ca (Teodor Gabriel Crainic), guido.perboli@polito.it (Guido Perboli), Walter.Rei@cirrelt.ca (Walter Rei), roberto.tadei@polito.it (Roberto Tadei)

of the problem as the *Variable Cost and Size Bin Packing Problem* (VCSBPP) to better distinguish it from the special case of the VSBPP.

Our contribution is twofold. First, we propose new and efficient heuristics for the VCSBPP. Most solution methods in the literature rely either on decomposition techniques [4, 5, 6] or on reformulations solved using commercial MIP software [7] and require significant computation times when applied to large instances. This computational effort makes them difficult to use in practice, planning settings in particular, where efficient, i.e., *fast* and *accurate*, solution methods are crucial to address the VCSBPP subproblems that must be solved repeatedly. The heuristic methods we propose successfully tackle the challenge of efficiency by using lower and upper bound techniques to select bins and pack items. Extensive numerical experiments were performed to support this claim, including on the special case of the VSBPP, for which we compare on a large set of instances the results of the new heuristics we propose to those of the best methods in the literature [7, 4]. This first numerical analysis indicates that the proposed heuristics find very good solutions extremely fast, even when addressing instances with up to 1000 items.

The second contribution we make in this paper is to provide a systematic numerical analysis of the impact on solution quality of the bin selection costs and of the correlations of these with the bin volumes. This analysis, initiated in the first computational phase mentioned above, is completed by experiments with a new set of instances specifically generated for the general VCSBPP setting. The results show that, indeed, these correlations matter and that the solution methods we propose, which are unbiased toward particular correlation values, perform better.

The paper is organized as follows. Section 2 discusses the interest of the VCSBPP setting with respect to applications, presents the model formulation, and reviews the methods proposed in the literature to address it. Lower bounds and the corresponding upper bound 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 Variable Cost and Size Bin Packing Problem

The first part of the section is dedicated to describing the general settings in which the VCSBPP is defined. We provide examples that both illustrate the usefulness of the problem and show how fixed costs may help formulate practical packing applications. We complete the section with the general formulation of the VCSBPP and the review of related literature.

2.1. Problem setting

Costs are an important attribute to take into account when addressing packing problems in many practical settings. Consider, for example, supply-chain management and the planning of the capacity required for the next cycle of activities (e.g., a year) of a logistics network. Warehousing and transportation activities are major components of such networks and the incorporation of packing formulations might significantly enhance their modeling, either in estimating the required capacity and its assignment to various product classes or in selecting the providers of logistics services and the types and quantities of vehicles (containers) to contract for with each of them. In all cases, the cost of the "bin", i.e., the unit of warehousing space or vehicle provided by each potential supplier, constitutes an important element when establishing the capacity plan.

Several factors influence the costs associated to the selection of bins, of which volume is only one. Thus, for example, the type of the "bin", be it an actual container or a unit of warehousing space is a determining factor. Several box types exist for each container size, from regular boxes and open tops to thermal containers and refrigerated ones. Obviously, the costs are different, but most may be used for the same product groups, given appropriate circumstances and substitution rules.

The availability of the space modeled by the bins is also an influencing factor. Turning again for illustration purposes to containers for maritime transportation, availability may vary with the particular physical, operational, and economic characteristics of the port of origin, as well as with the companies providing the containers and shipping services from that port [8].

One should also consider whether the fixed costs are used to model the prices of renting the bins or those of buying them. In the former case, the time period for which the container is to be used is an important factor that influences costs. When, on the other hand, a company is looking to buy containers, then obviously the overall condition of the units (new or used, in pristine condition or slightly damaged) becomes an important factor determining costs.

Considering the selection/use fixed cost as a relatively independent attribute associated with a particular bin, that may or may not be correlated to its volume, provides the means to address a much broader range of actual applications compared to the case studied up to now in the literature..

2.2. Formulation 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. Then, the VCSBPP model can be formulated as:

$$\min \quad z(y) = \sum_{j\in\mathcal{J}} c_j y_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{J}} x_{ij} = 1, \qquad \forall i \in \mathcal{I}, \tag{2}$$

$$\sum_{i\in\mathcal{I}} v_i x_{ij} \le V_j y_j, \quad \forall j \in \mathcal{J}, \tag{3}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, \quad \forall j \in \mathcal{J}, \tag{4}$$

$$y_j \in \{0,1\}, \qquad \forall j \in \mathcal{J}. \tag{5}$$

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 VCSBPP. In his Ph.D. thesis dedicated to packing and scheduling problems, Monaci presents lower bounds, as well as heuristic and exact solution methods for the VSBPP, i.e. the special case of the VCSBPP where the fixed costs of bins are equal to their volumes [4]. The methods presented in [4] take advantage of this correlation between bin volume and cost, and need to be modified to address the VCSBPP. Moreover, only aggregated results are presented, and private communications confirmed that the detailed ones are no longer available.

Seiden *et al.* [9] consider the on-line version of the problem and propose upper and lower bounds. Kang and Park [10] develop two greedy algorithms for the special case of the VCSBPP, 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 VCSBPP, with bin costs proportional to bin sizes, is proposed by Pisinger and Sigurd [5], together with lower bounds based on Dantzig-Wolfe decomposition and an exact branch-and-price algorithm. Alves and Valério de Carvalho [6] propose a series of strategies aimed at accelerating the column generation approach for the same problem.

Correia *et al.* [7] address the VSBPP and introduce bin selection costs that are strongly correlated to the bin volumes, while also displaying economies of scale. The authors discretize the formulation, propose valid inequalities to improve the quality of the lower bounds obtained from the linear relaxation of the resulting model, and analyze the quality of the lower bounds on a large set of instances (that we also use in the computational experiments of this paper).

3. Lower Bounds for the VCSBPP

We present a series of lower bounds for the VCSBPP, which 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. The first set of bounds is based on relaxations of the formulation (1)-(5) and knapsack optimization principles, while the last is based on column-generation ideas.

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 formulation (6)-(8) obtained by relaxing the integrality constraints (4) on item-to-bin assignment variables x_{ij} and aggregating the individual bin feasibility constraints (3).

This relaxation, originally proposed for the special case of the VCSBPP with bin fixed costs equal to their volumes [4], yields an optimal set of bins $\hat{y}^\star = (\hat{y}_1^\star, \hat{y}_2^\star, \ldots, \hat{y}_{|J|}^\star)$ with total capacity sufficient for the items considered, but with no guarantee of feasibility once individual bins are packed. Obviously, $z(\hat{y}^\star) \leq z^\star$.

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

The substitution (9)

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

applied to formulation (6)-(8) yields

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

which 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 [1]. Furthermore, any lower bound $LB(z(u))$ for problem (10)-(12) can also be used to produce a valid lower bound $LB(z(\hat{y})) = \sum_{j\in\mathcal{J}} c_j - LB(z(u))$ for problem (6)-(8). We note LB_1, the lower bound obtained from the original model (6)-(8).

LB_1 can be improved by considering that bins may be grouped according to the distinct values of their volumes [4]. 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 $M_k = \max\left\{ \sum_{i\in\mathcal{I}} v_i x_i \mid \sum_{i\in\mathcal{I}} v_i x_i \leq V_k,\ x_i \in \{0,1\}, \forall i \in \mathcal{I} \right\}$. 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

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

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

LB_1 can be further improved by considering that each item must be loaded. Define $M_{ik} = \max\left\{ \sum_{j\in\mathcal{I}\setminus i} v_i x_i \mid \sum_{i\in\mathcal{I}} v_i x_i \leq V_k - v_i,\ x_i \in \{0,1\}, \forall i \in \mathcal{I} \setminus i \right\}$ for each pair of item i and bin type k, corresponding to the maximum filling of a bin of type k when item i is loaded into it. Then, $t_i = \min_{k\in\mathcal{K}}\{V_k - M_{ik}\} - v_i$ represents the loss of volume due to the loading 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, requiring to solve a knapsack problem 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 non-decreasing values of their volumes v_i. The idea of the $\overline{M(p)}_{ik}$ approximation 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 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).

Both LB_2 and LB_3 dominate LB_1, but no dominance relationship can be defined for LB_2 and LB_3. Indeed, it is quite easy to define instances where LB_2 displays a better behavior than LB_3 and vice versa (see the example in the Annex). On the other hand, numerical experiments show that a new lower bound defined as $L_4 = \max\{LB_2, LB_3\}$ gives only marginal improvements compared to LB_2 or LB_3 separately.

We now apply to the general case of the VCSBPP the column generation lower bound introduced by Monaci [4] for the particular case of bin selection costs equal to their volumes.

Let us group the bins in $\mathcal{J}$ into types $t \in \mathcal{T}$, where two bins are defined to belong to the same type t if they have the same cost c_j and volume V_j. Define U_t as the number of available bins of type t. A *feasible loading pattern* for a bin of type t corresponds to a set of items that may be loaded into the bin satisfying all accommodation rules. Let $\mathcal{K}_t = \{k\}$ be the set of all feasible loading patterns for the bin type t, and $\mathcal{K} = \bigcup_{t\in\mathcal{T}} \mathcal{K}_t$. A feasible loading pattern k is described by a vector of indicator functions $a_k^i,\ k \in \mathcal{K}_t,\ t \in \mathcal{T}$, such that $a_k^i = 1$ if item i is accommodated by pattern k of bin type t, 0 otherwise. The cost of a pattern is the cost of the bin type.

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 VCSBPP may then be written as

$$\min \quad \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} c_k \lambda_k \tag{17}$$

$$\text{Subject to} \quad \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}_t} a_k^i \lambda_k = 1 \quad i \in \mathcal{I}, \tag{18}$$

$$\sum_{k\in\mathcal{K}_t} \lambda_k \le U_t \quad t \in \mathcal{T}, \tag{19}$$

$$\lambda_k \in \{0,1\} \quad k \in \mathcal{K}. \tag{20}$$

The lower bound is computed from the linear relaxation of formulation (17)-(20), which provides the means to implicitly deal with the large number of variables in the model through a column generation approach [11]. The procedure starts with a relatively small set of feasible patterns $\mathcal{P}$, corresponding to a restricted VCSBPP that we identify as *RP*, an then proceeds as follows:

1. Find an initial feasible solution to the VCSBPP and the corresponding set $\mathcal{P}$;
2. Solve the linear relaxation of *RP* and let L_{RP} be its optimal solution;
3. For each bin type $t \in \mathcal{T}$
   (a) Find the non-basic pattern variable $\lambda_{\bar{k}}$ of L_{RP}, $\bar{k} \in \mathcal{K}_t$, with the smallest reduced cost $r_{\bar{k}}$ among all non-basic pattern variables for type t;
   (b) If $r_{\bar{k}} < 0$, $\mathcal{P} = \mathcal{P} \cup \{\lambda_{\bar{k}}\}$;
   (c) Continue to the next bin type (go to 3a);
4. If $r_{\bar{k}} \ge 0$ for all bin types t, then stop with L_{RP} as the set covering lower bound for the VCSBPP;
   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 *RP* set covering formulation, μ_i, for constraints (18), and $\alpha_t \le 0$, for constraints (19). Let $\mathbf{A_k}$ be the column of a given pattern variable k for a bin of type t in model *RP*. Its reduced cost r_k is expressed as $c_k - [\mu\ \alpha]^T \mathbf{A_k}$, which becomes

$$r_k = c_t - \left[\mu^{\mathbf{T}}\ \alpha^{\mathbf{T}}\right] \mathbf{A^k} = c_t - \sum_{i\in\mathcal{I}} a_k^i \mu_i - \alpha_t.$$

We now define a column generation subproblem which, given a bin of type t, finds the non-basic pattern with the minimum reduced cost. Notice that the vector $\mathbf{A_k}$ defining a not yet generated pattern k of bin type t is not known, but may be expressed in terms of item-to-bin assignment variables x_i equal to 1 if item i belongs to the pattern, 0 otherwise (for simplicity of notation, we dropped the k index). Because the dual variables α_t, as well as the bin cost c_t, are constant for a given type t, to find the column with the minimum reduced cost we can simply solve the following

knapsack problem:

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{I}} \mu_i \, x_i \\ \text{Subject to:} \quad & \sum_{i \in \mathcal{I}} w_i x_i \leq W_t \qquad t \in \mathcal{T} \\ & x_i \in \{0,\ 1\} \qquad i \in \mathcal{I}. \end{aligned}$$

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 negative $r_{\overline{k}}$. Any feasible solution of the VCSBPP may be used to initialize the procedure, including the trivial solution obtained by loading each item into a different bin. More accurate heuristics are presented in Section 4. In the following, we refer to the objective function value of the optimal solution of the final continuous *RP* problem as LB_{CG}.

4. Heuristics for the VCSBPP

The heuristics we propose for the VCSBPP 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* (*BFD*) 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 [1]. The BFD we propose for the VCSBPP, named the *Adapted BFD* ($A-BFD$) heuristic, first sorts the items according to the non-increasing order of their volumes and, then, loads them sequentially; 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 VCSBPP, 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 an unused bin $k \in \mathcal{J},\ k \neq j$ such that $V_k \geq U_j$ and $c_k < c_j$ exists, the procedure 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*;
- 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.

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 $\mathcal{S}$

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 the procedures we propose by comparing the best results we obtain to those of the best methods proposed in the literature. Finally, we analyze the behavior of the methods in terms of efficiency and accuracy when problem attributes change, in particular, the correlation between the costs and volumes of the bins, and the volume distribution of the items.

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, LB_2, and LB_{CG} bounds were solved to optimality by means of the algorithm presented in [12], while the column generation in LB_{CG} was implemented in C++ by means of the CPLEX 12.1 solver.

5.1. Instance sets

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

Set1. Instances from [7]. 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.
- Instances with D and $D + 1$ bins are built, where D stands for the minimum number of bins for each type required to load all items.

Set2. Instances proposed in [4] and regenerated for [7], derived from classical bin packing instances for the item-volume distribution and relations to the bin volume [13]. 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.

Set3. New instances created with the goal of providing the means to explore the impact of the correlation between the selection fixed costs and the volumes of the bins. The new instances are thus characterized by various correlation strengths, as well as different compositions of the item set. The parameters of the new instances were also chosen to reflect city logistics and supply chain applications cases:
- Number of items in the set $\{100, 200, 500, 1000\}$.
- Bin capacity values in the set $\{50, 100, 120\}$. The values have been chosen to represent the typical volume ratios of ISO containers (20, 40 and 53 feet).
- Bin selection costs generated according to the following three rules:

  SC. As in [4], the costs of the bins are very strongly correlated to their volumes: they actually equal to their volumes.

  LC. The costs of the bins are loosely correlated with the volumes. In order to introduce an economy of scale, and following [7], the cost is defined as $\sqrt{V_j}$.

  R. For each bin volume, we generate three bin types. The first one has a cost of $\sqrt{V_j}$, the second $\sqrt{V_j}(1 + \delta)$, where δ is randomly generated between 0.05 and 0.3, and the third $\sqrt{V_j}(1 - \gamma)$, where γ is randomly generated between 0.05 and 0.3. This cost-generation scheme simulates the situation where different logistics operators are available. Thus, even though a basic price exists for each bin size, there could be changes of the price of each operator due to external factors.
- Item volumes. Following [14], the items are grouped into sets and, then, the item types are mixed in order to generate the item volume types. The items are grouped into three sets:

  G1. Big-sized items with volumes randomly generated in the interval $[20, 40]$.

  G2. Medium-sized items with volumes randomly generated in the interval $[15, 25]$.

G3. Small=sized items with volumes randomly generated in the interval $[5, 10]$.
Three item volume types are then generated mixing the item types in the following way:
T1. 80% of the items are in G2 and 20% in G3.
T2. 10% of the items are in G1, 75% in G2 and 15% in G3.
T3. 20% of the items are in G1, 70% in G2 and 10% in G3.
For each combination of the parameters, 10 instances are randomly generated.

5.2. Comparison of lower and upper bound versions

To compare the relative performance of the different versions of the upper bound procedures, 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 the varying p requires less that 10^{-2} in the worst case (most of it is 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 optimum.

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 according to the same idea described 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, favoring 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 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.

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

5.3. Comparison of lower and upper bounds with state-of-the-art algorithms

In the second part of this computational analysis, we present a comparison between the results obtained by the bounds we propose and the best results from the literature [4, 7]. The optimal solutions and the lower bound values obtained by Correia et al. are taken from the literature [7], while the results of the the column generation of Monaci [4] were obtained applying the proposed lower bound L_{CG}, the VSBPP being a special case of the VCSBPP (the detailed results of the original implementation are not available [4])

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 heuristic. We report in Table 4, however, the results of the instances of Set 1 not solved to optimality in [7]. 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 proposed by Monaci yielded a mean optimality gap of 0.23% on the instances of this set [4]. The proposed lower and upper bound procedures are thus performing very well on the instances introduced in [7], 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 %) from the optimal solutions yielded by the proposed lower bounds on instances in Set 2. The first and second columns 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 bounds LB_M by Monaci [4], LB_C by Correia et al. [7], and LB_1, LB_2, and LB_3 proposed in this paper. The last row of each instance type reports the mean deviations from the optimal values over all instances of that type.

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 while, for the other instances, it required 300 CPU seconds on average and up to 3000 CPU seconds for the largest instances (200 items) it addressed, on a 2.4 GhZ Pentium IV workstation.

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.

The column generation method of [4] 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. A more in-depth analysis of the performances of LB_M shows that this approach achieves best results on instances of type 3, characterized by the presence of large items resulting in the impossibility to load more than 2 items in the smallest bin. Moreover, the instances for which LB_M performs best are also characterized by a small number of items (25 and 50).. Consequently, such instances accept a small number of feasible patterns, which makes column generation based lower bounds efficient and accurate. Such instances are not representative of a large gamut of shipping and logistics applications, however, which makes LB_M less interesting as a general problem-solving tool.

The lower bound procedures we propose are the most 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. Actually, 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 to problem instances in a broad range of situations, and for

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

use in more complex problem-solving settings, e.g., simulations and optimization frameworks, where the VCSBPP appears as a subproblem in meta-heuristics or Branch & Bound schemes.

5.4. Analysis of the bin cost-volume relationships

The previous comparative analysis of the performance of the proposed lower and upper bounds demonstrated their efficiency and accuracy on the instances of sets 1 and 2. These instances present particular characteristics in terms of bin selection costs, which are very tightly correlated to the bin volumes. The goal of of this part of the computational study is to investigate the impact on the behavior of the methods we propose of different problem characteristics, in particular, the correlations between the costs and volumes of the bins.

IT	**BIN**	LB_1	LB_{CG}
T1	**SC**	0.33	0.44
	LC	0.78	1.36
	R	1.17	1.46
T2	**SC**	0.53	0.61
	LC	0.58	1.29
	R	1.35	1.61
T3	**SC**	0.51	0.60
	LC	0.49	1.16
	R	1.31	1.63

Table 6: Gaps (%) of LB_1 and LB_{CG} relative to $Comp_{BFD}$

Table 6 displays the comparative results of the lower bounds LB_1 and LB_{CG} relative to those of the $Comp_{BFD}$ heuristic. We do not report the results of bounds LB_2 and LB_3 because their gaps relative to LB_1 are small and do not change the results of the analysis. We report in Columns 1 and 2 the bin selection cost and the item distribution type. Columns 3 and 4 display the mean gap between LB_1 and LB_{CG} with respect to $Comp_{BFD}$, respectively, computed

as $(UB - LB)/LB$. Each cell gives the mean gap over the 40 instances characterized by the same bin cost - volume combination.

The results show that LB_1 displays the best performance for all groups of instances. This is not surprising, column generation approaches being more effective when the number of possible equivalent columns (item loading patterns) is somewhat limited, which holds true for Set 2, but not for the instances of Set 3 used in this experiment. The overall mean gap is about 1% in the worst case. The instances with the highest gap are in set R, characterized by a low correlation between volume and cost of the bins. No difference in solution quality can be noticed while changing the item volume type. When the mix of the item sizes changes, the gap remains almost constant for the three distributions.

An interesting question is whether the gap is mainly given by the lower or the upper bounds. To address this issue, and inspired by the studies in [15] on the instances of Set 2 where the column generation yielded the optimal patterns for the majority of instances, we defined UB_{CG}, an upper bound procedure that starts from the model (17)-(20) with the patterns identified while computing LB_{CG}. All the instances of Set 3 were thus solved to optimality using the CPLEX 12.1 solver. We then computed the gaps of $Comp_{BFD}$ and UB_{CG} with respect to LB_1.

We do not report the detailed results for $Comp_{BFD}$ and UB_{CG} in order not to overload the paper. Globally, however, the solution found by UB_{CG} is equal to that of $Comp_{BFD}$ in 27 instances over 360, reducing the gap with respect to LB_1 from 0.60% to 0.59%. We thus infer that the results of $Comp_{BFD}$ and UB_{CG} are generally "close" and, thus, even though no formal proof of the optimality of the solutions $Comp_{BFD}$ was obtained, these comparisons suggest that the gap is mainly due to the lower bound LB_1.

IT	BIN	USED	SAT
T1	**SC**	2.4	0.1
	LC	1.8	0.3
	R	4.5	1.7
T2	**SC**	2.3	0.2
	LC	2.0	0.4
	R	4.5	1.5
T3	**SC**	2.1	0.2
	LC	1.9	0.3
	R	4.6	1.9

Table 7: Number of bin types used and saturated

Another interesting issue is related to the number of bin types used in the final solution, relative to variations in the correlation between the selection costs and the volumes of the bin types. Table 7 displays this characteristic of the solutions produced by $Comp_{BFD}$. Columns 1 and 2 report the bin cost and the item distribution types, respectively, while Columns 3 and 4 display the mean number of bin types used and saturated, where a bin type is considered saturated if all the bins belonging to that type are used in the given solution. As indicated by these results, the number of bin types used is affected by the bin cost distribution relative to the bin volume. Indeed, while the number of used and saturated bin types is almost the same for types SC and LC, the mix changes a lot for the instances in R, when several bin types are characterized by the same costs but different volumes. It is also noteworthy that the number of saturated bin types increases with the variation in cost versus volume relation, indicating the capability of the $Comp_{BFD}$ procedure to select bins with a small cost per unit of volume.

Most previous analyzes were performed with respect to computing times and solution quality. In this respect, LB_1 displays an excellent behavior, its computational effort being determined by the resolution of a single Knapsack instance. We may thus present a comparative analysis of LB_1 and LB_{CG} based on the number of columns the methods generate, a measure that may be used to predict their behavior for larger problem instances. As indicated earlier on, the generation of each column produced by LB_{CG} requires solving a Knapsack instance, the overall computational effort of the procedure being dominated by the resolution of these instances. According to our results, the main parameter affecting the number of columns generated by LB_{CG} is the number of items. Figure 1 thus reports the relationship between the number of items and the number of columns generated by LB_{CG}. It should be noticed that, already for instances with 500 items, the procedure generates about 2000 columns, number that raises to about 7000 for 1000-

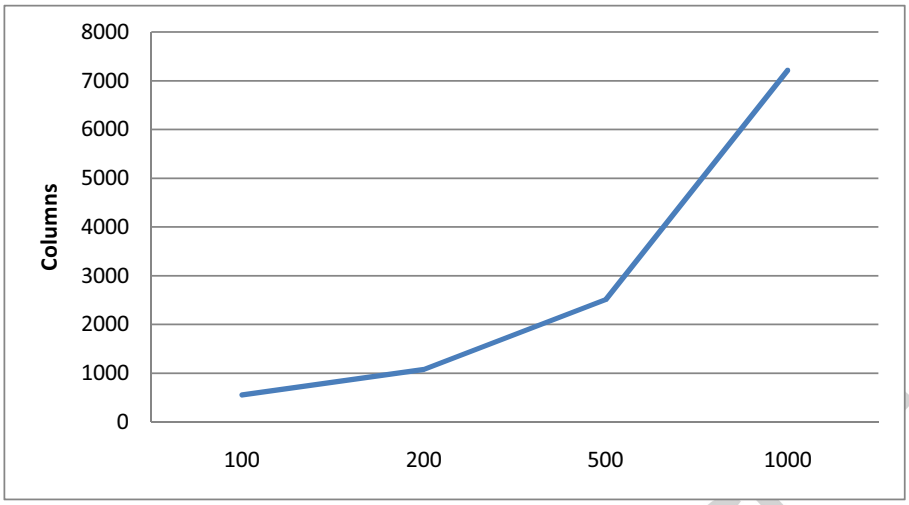

Figure 1: Columns generated by LB_{CG} with respect to the number of items

item instances. This implies a computational effort for LB_{CG} three order of magnitude larger than that of LB_1 and points to the interest of the heuristics proposed in this paper to address large instances of the Variable Cost and Size Bin Packing Problem.

It is revealing to analyze these results in the light of the very small number of columns, 46, needed by LB_M for the instances in Set 2. Recall that LB_M is equivalent to the lower bound LB_{CG} for the VSBPP problem. Recall also that the instances of Set 2 are characterized by important proportions of large items, which reduces considerably the number of feasible patterns and, thus, of generated columns, and makes them less representative of actual applications. Focusing then on the results of the various procedures on the instances of Sets 1 and 3, one may conclude that the proposed methodology, LB_1 in particular, is both more efficient and more accurate than a column generation-based approach, such as LB_M and LB_{CG}, for large-sized logistics applications.

6. Conclusion

We introduced the Variable Cost and Size Bin Packing Problem, an extension of the classical Variable Size Bin Packing Problem, characterized by both physical and economical attributes, bin volumes (capacity) and selection fixed costs, in particular. We proposed a solution methodology based on new upper and lower bounds for the VCSBPP. These procedures proved to be extremely efficient on a large set of problem instances with up to 1000 items and varying correlations between the bin volumes and selection costs. Indeed, compared to state-of-the-art methods, the proposed methodology yielded solution with an accuracy of more than 99%, with a computational effort several order of magnitude smaller.

We also presented, for the first time in the associated literature, a comprehensive study of the impact on the solution-method performance of the correlation between the cost and volume of the bin types, as well as of how the structure of the solutions changes with it. The results show that, indeed, this correlation matters and that the solution methods we propose, which are unbiased toward particular correlation values, offer a better performance.

Acknowledgments

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

While working on this project, the first author was the NSERC Industrial Research Chair in Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université

de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. This project has been partially supported by the Ministero dell'Istruzione, Universitá 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

[1] S. Martello, P. Toth, Knapsack Problems - Algorithms and computer implementations, John Wiley & Sons, Chichester, UK, 1990.
[2] H. Dyckhoff, A Typology of Cutting and Packing Problems, European Journal of Operational Research 44 (1990) 145–59.
[3] G. Wäsher, H. Haußner, H. Schumann, An improved typology of cutting and packing problems, European Journal of Operational Research 183 (2007) 1109–30.
[4] M. Monaci, Algorithms for packing and scheduling problems, Ph.D. thesis, Universit di Bologna, 2002.
[5] D. Pisinger, M. Sigurd, The two-dimensional bin packing problem with variable bin sizes and costs, Discrete Optimization 2 (2005) 154–67.
[6] C. Alves, J. M. Valérion de Carvalho, Accelerating column generation for variable sized bin-packing problems, European Journal of Operational Research 183 (2007) 1333–52.
[7] I. Correia, L. Gouveia, F. Saldanha-da-Gama, Solving the variable size bin packing problem with discretized formulations, Computers and Operations Research 35 (2008) 2103–13.
[8] N. Limão, A. J. Venables, Infrastructure, geographical Disadvantage, Transport Costs, and Trade, The World Bank Economic Review 15 (3) (2001) 451–79.
[9] S. S. Seiden, R. Van Stee, L. Epstein, New bounds for variable-sized online bin packing, SIAM Journal on Computing 32 (2) (2003) 455–69.
[10] J. Kang, S. Park, Algorithms for the variable sized bin packing problem, European Journal of Operational Research 147 (2003) 365–72.
[11] F. Vanderbeck, Computational study of a column generation algorithm for bin packing and cutting stock problems, Mathematical Programming 86 (1996) 565–94.
[12] D. Pisinger, A minimal algorithm for the 0-1 knapsack problem, Operations Research 45 (1997) 758–67.
[13] F. Vanderbeck, Computational study of a column generation algorithm for bin packing and cutting stock problems, Mathematical Programming (1999) 565–94.
[14] S. Martello, D. Pisinger, D. Vigo, The Three-Dimensional Bin Packing Problem, Operations Research 48 (2) (2000) 256–67.
[15] M. M. Baldi, T. G. Crainic, G. Perboli, R. Tadei, The Generalized Bin Packing Problem, Publication CIRRELT-2010-21, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2010.

Annex - Illustration of relationships between lower bounds

As stated in Section 3, no dominance relation may be defined between LB_2 and LB_3. This is an illustration of this statement through two instances for which LB_2 displays a better behavior than LB_3 and vice versa, respectively.

Two bin types are defined for both instances: 12 bins with cost 1 and volume 100 for type 1 and 1 bin with cost 2 and volume 105 for type 2. The two instances have the same number of items but differ in the characteristics of some of them. In instance I_1, we have 10 items with volume 60, 10 items with volume 30, and 2 items with volume 50. The mix for instance I_2 is the same except for the last group where the items with volume 50 are replaced by items with volume 45.

It is clear that, for both instances, bins of type 2 will not be included in the optimal solution or any lower bound, as bins of type 1 are less costly and offer sufficient capacity to accommodate all the items.

In I_1, LB_2 is not able to decrease the best filling of bins of type 1, due to the presence of the two items with volume 50, while LB_3 puts $t_i = 10$ for all the items with volume equal to 60. Thus their volume changes from 60 to 70 and the values of the lower bounds are $LB_2 = 10$ and $LB_3 = 11$. Performing the same computations for I_2, we have that for LB_2 the best filling ratio of bins of type 1 decreases from 100 to 90, yielding $LB_2 = 11$. In this case LB_3 is not able to improve LB_1. In fact, for any item, we are able to find a combination with the remaining items that fills the bin of type 2. Thus $t_i = 0$ for all the items and $LB_3 = LB_1 = 10$.