Continuous Lower Bound for the Variable Sized Bin-Packing Problem

Maiza Mohamed, Radjef Mohammed Said, Lakhdar Saïs

9^{th} *International Conference of Modeling, Optimization and Simulation - MOSIM'12*
June 6-8, 2012 - Bordeaux - France
"Performance, interoperability and safety for sustainable development"

CONTINUOUS LOWER BOUND FOR THE VARIABLE SIZED BIN-PACKING PROBLEM

Mohamed MAIZA[(1)(2)] **Mohammed Said RADJEF**[(1)(3)] **Lakhdar SAIS**[(2)]

[(1)]Laboratoire des Mathématiques Appliquées, École Militaire Polytechnique, 16111 Bordj-El-Bahri, Alger-Algérie.
[(2)]Centre de Recherche en Informatique de Lens CNRS-UMR8188, Faculté Jean Perrin, 62307 Lens-France.
[(3)]Laboratoire Modélisation et Optimisation des Systèmes, Université de Béjaïa, 06000 Béjaïa-Algérie.

maiza@cril.fr radjefms@yahoo.fr sais@cril.fr

ABSTRACT: *In this paper we consider the Variable Sized Bin Packing Problem (VSBPP), a generalization of the classical one dimensional Bin-Packing Problem (BPP), where a set of items must be packed into a set of heterogeneous bins characterized by different volumes and when the cost of unit size of each bin does not increase as the bin volume increases. The objective is to minimize the total cost of bins needed to store given items, each item with some space requirement. We develop a new continuous lower bound inspired from a generalization of the well known continuous lower bound for the classical BPP. Then, we test the performances of the proposed procedures by means of computational results on benchmarks instances taken from the literature. Theses results show the relevance of our proposition.*

KEYWORDS: *Bin-packing problem, Continuous bound, Monotonicity constraint.*

1 INTRODUCTION

The *classical bin-packing problem* (BPP) is a well-studied combinatorial optimization problem requiring the assignment of a given set of items to a minimum number of identical bins of fixed capacity. This combinatorial optimization problem belongs to the class of *NP-hard* problems in the strong sense (Garey & Johnson 1979). An excellent survey of this problem can be found in Chapter 2 of (Coffmann, Garey & Johnson 1997). The influence and importance of this problem are witnessed by the fact that it has spawned off various areas of research, including the fields of algorithms and approximation algorithms. In the natural generalization of the BPP known as *Variable Sized Bin-Packing Problem* (VSBPP), bins are grouped by category and each one contains infinite identical bins with the same cost and capacity, the objective function becomes to minimize the total cost of bins used to pack all items. This particularity arises in many practical applications such as cutting-stock problems, loading truck problems, assignment of process to processors, machine and telecommunication scheduling.

In this paper, we investigate the off-line one dimensional VSBPP when the **monotonicity constraint** was verified (Kang & Park 2003) (*the unit cost of each bin does not increase as the bin size increases*). Each category of bins can be seeing with an infinite number of identical bins. The objective is to minimize the total cost of bins used to store given items where each item must be assigned in at least one bin. In other words, the greatest item cannot exceed the largest bin.

The literature on VSBPP bounding algorithms is quite rich. This problem was first investigated by Friesen and Langston (Friesen & Langston 1986). The authors gave three approximation algorithms with worst-case performance bounds of 2, 3/2 and 4/3. Then, Murgolo (Murgolo 1987) gave a fully polynomial time asymptotic approximation scheme for the problem. Monaci (Monaci 2002) solved the VSBPP by means of branch and bound algorithm applied on instances which consider 3 to 5 bin-types and at most 500 items. Other methods have been used to solve this problem; we see a discretized formulation of Correia *et al* (Correia, Gouveia & da Gama 2008) where authors make tighter the mathematic formulation of the problem using suitable valid inequalities. Kang and Park (Kang & Park 2003) described and analyzed two greedy algorithms based on well-known BPP algorithms (FFD and BFD) where authors ensure the optimal solution if the sizes of items and bins are divisible, respectively. For other variants of the problem, it is worth to note that Epstein and Favrholdt (Epstein & Favrholdt 2002) featured the VSBPP with maximizing the number of packed items in a fixed number

of bins. Recently, Righini *et al.*(Righini, Bettinelli & Ceselli 2010) solved the VSBPP with minimum filling constraint using a branch and price algorithm, Crainic *et al.*(Crainic, Perboli, Rei & Tadei 2011) proposed a solution methodology based on upper and lower bounds for the VCSBPP (Variable Cost and Size Bin-Packing Problem) where the cost variation does not depend on volume.

To our knowledge, no general continuous lower bounds were proposed before for the VSBPP. So, a formal definition and a basic formulation for this problem are given in the next section (section 2). In Section 3 we define the well known simplest continuous lower bound for the classical BPP. Then, we drawing it for the cases of two categories of bin. After that, we generalize our study to the case where the number of bin categories is greater than two. The performances of proposed lower bound are tested in Section 4 by means of computational results on benchmarks instances taken from the literature. Finally, we close this paper with concluding remarks.

2 Problem formulation

Consider a set $\mathrm{I} = \{1,2,...,n\}$ of n items, a set $\mathrm{K} = \{1,2,...,m\}$ of m bin-types, and a set $\mathrm{J}_k = \{b_1^k, b_2^k, ..., b_{n_k}^k\}$ of a fixed and sufficient number n_k of identical bins for each bin-type $k \in \mathrm{K}$. Let H_k,c_k be the capacity and the cost of the k^{th} bin-type respectively. Let $h_i \in \mathbb{N}^+$ be the size of item i where $h_i \leq \max_k(H_k)$ $\forall i \in \mathrm{I}$. Without loss of generality we assume that $h_1 \geq h_2 \geq ... \geq h_n$ and $H_1 \geq H_2 \geq ... \geq H_m$. The VSBPP consists in assigning each item i to one bin in order to minimize the total cost of used bins. This problem is a natural generalization of BPP where only one bin-type is considered. In this paper, we investigate the VSBPP when the unit cost $u_k = c_k/H_k$ of k^{th} bin-type does not increase with the size of bin-type, then $u_1 \leq u_2 \leq ... \leq u_m$. The VSBPP can be modeled as an integer linear program (ILP) by introducing two sets of binary variables: x_{ij_k} equal to 1 if item i is assigned to bin j of bin-type k, 0 otherwise; y_{j_k} taking value 1 if bin j of k^{th} bin-type is used, 0 otherwise. Then we have the following ILP:

$$\min\Big(\sum_{k=1}^{m}\sum_{j_k=1}^{n_k} c_k \cdot y_{j_k}\Big) \tag{1}$$

$$\sum_{i=1}^{n} h_i \cdot x_{ij_k} \leq H_k \cdot y_{j_k} \qquad k = 1,...,m; j_k = 1,...,n_k \tag{2}$$

$$\sum_{k=1}^{m}\sum_{j_k=1}^{n_k} x_{ij_k} = 1 \qquad i = 1,...,n. \tag{3}$$

$$x_{ij_k} \in \{0,1\}; y_{j_k} \in \{0,1\} \qquad \forall i \in \mathrm{I}, \forall k \in \mathrm{K} \tag{4}$$

The objective function to be minimized (1) represents the cost of the bins used to pack all the items. Constraint (2) indicates that the amount packed in each used bin does not exceed the capacity of this bin. Constraint (3) ensures that each item i has to be packed, whereas the constraint (4) represents the integrality constraint. In our investigation, the following data that correspond to the above problem are known and available at the beginning of assignment:

- the quantity and sizes of items;
- the number of bin-types with the quantity and capacity of each one;
- the cost or unit cost of each bin-type.

3 Continuous lower bound (CLB)

Continuous lower bound is a relaxing bound considered where items can be assigned to bins in fractional way or where items are considered like item of unit size. This problem is combinatorial for $m > 1$ where m represents the number of bin-types, and it is also *NP-hard*. The optimal solution of this problem called in what follows *'optimal continuous solution'* can be obtained by solving the following integer linear programming (ILP):

$$\min \sum_{k=1}^{m} c_k \cdot x_k \tag{5}$$

$$\sum_{k=1}^{m} x_k H_k - \sum_{i=1}^{n} h_i \geq 0 \tag{6}$$

$$0 \leq x_k \leq \left\lceil \frac{\sum_i h_i}{H_k} \right\rceil \quad k = 1,...,m. \tag{7}$$

where x_k is an integer represents the number of bins used from category k.

3.1 CLB for the classical BPP

For the classical BPP ($m = 1$), the objective is to minimize the number of identical bins needed to store all items. Therefore, the simplest allowed lower bound corresponds to the so called continuous lower bound reported by Martello *et al.*(Martello & Toth 1990b) and denoted CLB_{BPP}. This lower bound computed as the rounding up of the sum of the item sizes divided by the bin capacity, by solving the previous ILP (5–7):

$$x_1 = \mathrm{LB}_{BPP}(\mathrm{I}) = \left\lceil \frac{\sum_{i \in \mathrm{I}} h_i}{H} \right\rceil \tag{8}$$

In terms of cost, when we have to minimize the total cost of used bins equation (8) becomes:

$$\mathrm{LB}_{BPP}(\mathrm{I}) = c \cdot \left\lceil \frac{\sum_{i\in \mathrm{I}} h_i}{H} \right\rceil \tag{9}$$

3.2 CLB for the Two-VSBPP

Let's now consider the case of only two categories of bins with pairs (H_1,u_1) and (H_2,u_2) representing the capacity and unit cost respectively. We called in the sequel this cases Two-VSBPP. Without loss of monotonicity constraint imposed in this paper ($H_1 \geq H_2$ and $u_1 \leq u_2$), it is obvious that, if $c_1 \leq c_2$ the last bin-type is dominated and should consequently not considered. Then, we consider only the first bin-type which represents the cheapest bins and the lowest unit cost bin. Therefore, the corresponding lower bound is given by formula (9), by considering c and H as the cost and the capacity of the first category of bins (c_1 and H_1).

3.2.1 cases of $c_1 > c_2$

A more important contribution concerns the cases of $c_1 > c_2$. Indeed we suggest a tighter continuous lower bound. Under the assumption that items can be split, we will consider in the sequel, a continuous quantity S which initially represents the total size of items I. The optimal packing is performed by using only bins of category 1 (each bin is completely filled). In other words, if S exactly divides the capacity of the lowest unit cost bin-type, then, continuous lower bound is directly given by considering only the first category of bins. Otherwise, the remaining quantity which is not yet considered (denoted Q in what follows) can be assigned according to the cost and the capacity of the second category of bins.

We consider a positive integer value α ($\alpha \in \mathbb{Z}^+$) representing the quotient of both costs. Then, we have:

$$\begin{aligned} \alpha = \left\lfloor \frac{c_1}{c_2} \right\rfloor \quad &\Rightarrow \quad \alpha \leq \frac{c_1}{c_2} \\ &\Rightarrow \quad c_1 \geq \alpha \cdot c_2 \qquad (10) \\ &\Rightarrow \quad u_1 \cdot H_1 \geq \alpha \cdot u_2 \cdot H_2 \\ &\Rightarrow \quad H_1 > \alpha \cdot H_2 \qquad (11) \end{aligned}$$

In order to minimize the total cost of the used bins, it is obvious from formula (10) and (11) that using α bins of category2 requires a lower cost than using one bin of category1. Consequently, if the remaining quantity Q of the sum S is less than αH_2, it is more advantageous to restrict to the use of bins of category2.

Moreover, from 10 we have:

$$c_1 + c_2 \geq (\alpha + 1) \cdot c_2 \tag{12}$$

Hence, we conclude from formula (12) that if the remaining quantity Q of S exceeds the capacity of the first category of bins and at the same time it is less than $(\alpha+1)H_2$, then it is more advantageous to use $(\alpha+1)$ bins of category 2 instead of combining one bin of category1 and one bin of category2.

Consequently, our continuous lower bound helps in continuously filling up bins of category1 until we get the remaining quantity Q less or equal to $\max(H_1;(1+\alpha)H_2)$. So, continuous lower bound of remaining quantity Q will be given according to one of the following cases:

$$\mathrm{CLB}(Q) = \begin{cases} c_2 \cdot \lceil \frac{Q}{H_2} \rceil & \text{if } 0 < Q \leq \alpha H_2 \\ c_1 & \text{if } \alpha H_2 < Q \leq H_1 \\ (\alpha+1) \cdot c_2 & H_1 < Q \leq (\alpha+1) H_2 \end{cases} \tag{13}$$

where $Q \leq \max(H_1;(1+\alpha)H_2)$.

The main steps of continuous lower bound calculation for a given quantity S can be summarized in algorithm 1.

Algorithm 1 Two-VSBPP continuous lower bound whith $c_1 > c_2$.

Procedure $\mathrm{CLB}_{Two-VSBPP}(S,\alpha)$

```
1:  CLB = 0; V = 0;
2:  while S > max(H1;(1+α)H2) do
3:     CLB = CLB + c1;
4:     S = S − H1;
5:  end while
6:  if (S ≤ max(H1;(1+α)H2)) and (S ≠ 0) then
7:     if (0 ≤ S ≤ αH2) then
8:        V = c2 · ⌈S/H2⌉;
9:     else
10:       if αH2 ≤ S ≤ H1 then
11:          V = c1;
12:       else
13:          V = (1+α) · c2;
14:       end if
15:    end if
16: end if
17: CLB = CLB + V;
18: return CLB;
```

For illustration, given two categories of bins with pairs (6, 1) and (4, 1.1) respectively, so $\alpha = 1$, and given a set I of five items with sizes 5, 5, 5, 3, 2, so, the total items size S is equal to 20. The continuous

lower bound can be calculated progressively in three iterations by using two bins of category1 and two bins of category2 with a total cost of 20.8;

3.2.2 cases of $c_1 > c_2$ and $H_2 > \frac{1}{2}H_1$

A particular case can be shown when $H_2 > \frac{1}{2}H_1$. For that, we define two subsets of mutually complementary items $\mathrm{I}^{'}$ and $\mathrm{I}^{''}$ are defined such as:

$$\mathrm{I}^{'} = \{i \in \mathrm{I} : H_2 < h_i \leq H_1\} \text{ and } \mathrm{I}^{''} = \mathrm{I} \setminus \mathrm{I}^{'}$$

Obviously, we cannot assign more than one item of $\mathrm{I}^{'}$ to the same bin of category1. Moreover, each item of $\mathrm{I}^{'}$ will be loaded to a different bin of category1. Thus, we suppose that only items of $\mathrm{I}^{''}$ are used in a continuous way in order to completely fill bins of category1 initially occupied by items of a subset $\mathrm{I}^{'}$. Then, the remaining quantity Q_r of $\mathrm{I}^{''}$ which will be assigned, can be defined as:

$$Q_r = \max(0; S - (|I^{'}| \cdot H_1)) \tag{14}$$

So, the total cost of the used bin becomes the cost of a single bin needed to fit each item of subset $\mathrm{I}^{'}$, plus the cost obtained by application of continuous lower bound to the remaining quantity Q_r of a subset $\mathrm{I}^{''}$. So, algorithm 1 becomes:

Step 0 Separate out subsets $\mathrm{I}^{'}$ and $\mathrm{I}^{''}$ from initial set of items I;

Step 1 Put $\mathrm{CLB}^{'} = c_1 \cdot |\mathrm{I}^{'}|$ and calculate the quantity Q_r using formula(14);

Step 2 Apply Procedure $\mathrm{CLB}_{Two-VSBPP}(Q_r, \alpha)$ to obtain continuous lower bound $\mathrm{CLB}^{''}$ corresponding to the remaining quantity Q_r;

Step 3 Continuous lower bound $\mathrm{CLB}_{Two-VSBPP}(\mathrm{I})$ will be obtained by summing the both costs $\mathrm{CLB}^{'}$ and $\mathrm{CLB}^{''}$;

For illustration we consider as well the previous example, only items of size 5 are assigned to bins of category1. Application of procedure $\mathrm{CLB}_{Two-VSBPP}(Q_r, \alpha)$ to remaining quantity $(Q_r = 2)$ leads to the use of one bin of category2. Then, the total cost (continuous lower bound) becomes 22.4 and corresponds to the use of three bins of category1 and one bin of category2.

3.3 CLB for the general VSBPP

We now generalize our study to the case where the number of bin categories is greater than two. We suppose that $c_1 > c_2 > ... > c_m$ and we also suppose that $\alpha_j : \alpha_j \in \mathbb{Z}^+$ is the positive integer value represents the cost quotient between both bins of category j and bins of subsequent category $j+1$, with $j = \{1, ..., m-1\}$.

Starting from the two first categories of bins, we will refer the same reasoning as in the previous case (case of two categories of bins). In each iteration we save the obtained cost solution. Then, we try to improve a continuous lower bound by considering the remaining quantity to be assigned to the second category $j+1$, as an input quantity in the next iteration which consider the current $j+1$ and the subsequent $j+2$ category. In this way, our continuous lower bound $\mathrm{GCLB}_{VSBPP}(\mathrm{I})$ consists of iteratively running procedure $\mathrm{CLB}_{VSBPP}(j, j+1, Q, \alpha_j)$ for two successive categories of bins $(j, j+1)$, where in each iteration only the given quantity assigned to the last category of bins is considered for the next iteration. In other words, starting from a quantity S which is a sum of size of items, our procedure attempts to find a better assignment of remaining quantity provided in each iteration. Continuing this procedure until either $j = m$ or $S = 0$, we have some feasible solutions at hand and the best solution among them is selected as the final solution.

The general $\mathrm{GCLB}_{VSBPP}(\mathrm{I})$ can be implemented iteratively or recursively. An iterative implementation is shown in Algorithm 2.

Algorithm 2 VSBPP continuous lower bound in the general cases.

Procedure $\mathrm{GCLB}_{VSBPP}(S)$

1: $GLB(0) \leftarrow 0$;
2: $V(0) \leftarrow 0$;
3: $j \leftarrow 1$;
4: **while** $(j \neq m-1)$or $(S \neq 0)$ **do**
5: Call Procedure $\mathrm{CLB}_{Two-VSBPP}(j, j+1, S, \alpha_j)$ and get $CLB(j)$ and $V(j)$;
6: $S \leftarrow$Quantity assigned to category $(j+1)$;
7: $GLB(j) \leftarrow GLB(j-1) - V(j-1) + CLB(j)$
8: $j \leftarrow j+1$;
9: **end while**
10: **return** $\min(GLB(j))$

In the general case where it is not necessary that $c_1 > c_2 > ... > c_m$, we initially considered the first bin-type for the continuous lower bound calculation, then, only the next lowest cost category compared from the last taken will be taken into account.

For illustrates this procedure, let us take an example with 5 bin-types with triplet capacity, unit cost and cost are respectively (200, 1, 200); (180, 1.5, 270);

(100, 1.7, 170); (90, 2, 180); (50, 3, 150). Hence, the continuous lower bound will be calculated by procedure of algorithm 2 with considering only the first, the third then the fifth bin-type, because $c_1 > c_3 > c_5$.

4 Computational results

Proposed continuous lower bound procedure was coded in *C++* and ran on an Intel Xeon 2.0 GHz with 2GB of RAM under a Windows XP operating system. We consider two data-sets called in what follows *DS*1 and *DS*2 respectively. Theses data-sets are inspired from the VSBPP literature, generated by Monaci (Monaci 2002) and involves instances with up to 500 items and 5 categories of bins. These data-sets involve instances with $n = 25, 50, 100, 200$ and 500 items, which are split into six groups according to the number of bin-types (3 or 5) and the intervals in which the item sizes vary ($[0-100]$, $[20-100]$, $[50-100]$). Each combination: number of items, number of bin-types and item size distribution, contains 10 instances. Consequently, we get a total of 300 instances for each data-set (*DS*1 and *DS*2). Moreover, we fixed the unit cost in such that we verify the condition $c_1 > c_2 > ... > c_m$ and we assume that $u_{i+1} = u_i + 1/10$ with $u_1 = 1$. The difference between data-sets *DS*1 and *DS*2 lies in the considered bin-type capacities (H_j). For the first data-set, we considered 3 bin-types with capacities $150, 120, 100$ respectively; or 5 bin-types with capacities $150, 120, 100, 80, 60$ respectively. Whilst, in the *DS*2 instances, in addition to the monotonicity criterion, we also assume that at least there is an $\alpha_j > 1$. Then, we consider 3 bin-types with capacities $250, 110, 60$ respectively; or 5 bin-types with capacities $300, 250, 100, 80$ and 20 respectively.

We summarized the computational results of the *CLB* in tables 1 and 2 using data-sets *DS*1 and *DS*2 respectively. These results are given as averages of every three intervals of items size, i.e. each line contains average values over 30 instances and evaluated up to the optimal continuous solution obtained by solving the ILP (5- 7) using an exhaustive enumeration to test all possible combinations. A limiting time of ten seconds is imposed on calculation of the optimal continuous solution of any instance and drawn characters refer to cases where this time limit was reached.

Both tables 1 and 2 contain three blocks. The first contains the number of items and the number of bin-types respectively. The two others give the results referred to both the simplest lower bound *SLB* calculated with formula (9) and our continuous lower bound *CLB*. Each block contains three columns showing: 1)-the average deviation *dev%* computed

Table 1: Computational results of continuous lower bound using data-set *DS*1.

		SLB			CLB		
n	m	*dev.(%)*	*opt.*	*ct.(µs)*	*dev.(%)*	*opt.*	*ct.(µs)*
25	3	1.97	*0*	*9.33*	0.68	*15*	*8.48*
	5	1.91	*0*	*9.23*	0.32	*18*	*9.44*
50	3	1.10	*0*	*9.29*	0.54	*13*	*8.73*
	5	0.97	*0*	*9.07*	0.28	*18*	*12.60*
100	3	0.56	*0*	*9.53*	0.31	*15*	*9.12*
	5	0.52	*0*	*9.55*	0.10	*22*	*9.99*
200	3	0.25	*0*	*9.88*	0.11	*20*	*9.98*
	5	-	-	*9.78*	-	-	*10.76*
500	3	0.09	*2*	*10.55*	0.03	*25*	*12.62*
	5	-	-	*10.54*	-	-	*13.34*
Average		0.92	*2/240*	*9.67*	**0.30**	*146/240*	*10.51*

Table 2: Computational results of continuous lower bound using data-set *DS*2.

		SLB			CLB		
n	m	*dev.(%)*	*opt.*	*ct.(µs)*	*dev.(%)*	*opt.*	*ct.(µs)*
25	3	2.06	*0*	*9.37*	0.42	*16*	*8.67*
	5	2.20	*0*	*9.36*	0	*30*	*9.53*
50	3	1.26	*0*	*9.29*	0.39	*14*	*8.73*
	5	1.22	*0*	*9.08*	0.05	*22*	*12.6*
100	3	0.60	*0*	*9.56*	0.12	*18*	*9.12*
	5	0.57	*0*	*9.54*	0	*30*	*9.99*
200	3	0.25	*1*	*9.65*	0.06	*26*	*9.77*
	5	-	-	*9.91*	-	-	*11.03*
500	3	0.11	*1*	*10.53*	0.02	*28*	*12.61*
	5	-	-	*10.54*	-	-	*13.49*
Average		1.03	*2/240*	*9.68*	0.13	*184/240*	*10.55*

as $((C^* - CLB)/C^*) \cdot 100$, where C^* is the optimal continuous solution obtained by solving the ILP and CLB is the solution obtained by the corresponding lower bound; 2)-the number $opt.$ which gives the number of times where the optimal solution has been reached; 3)-the average computing time $ct.$ of the corresponding solution, in microseconds.

Computational results indicate that the same performances are provided for both $DS1$ and $DS2$. In the two features, the *CLB* is more powerful than the *SLB* and a very close solution to optimality can be found in very reasonable time. For the $DS1$ instances, the deviation of solutions is less than 0.7% in the worst case and of 0.3% in average, while, for the $DS2$ instances it is less than 0.5% in the worst case and of 0.13% in average. In addition, *CLB* ensures that more than 60% of the $DS1$ instances and more than 76% of the $DS2$ instances lead to the optimal solution. Computing time (in microseconds) indicates the speed performance of both *CLB* and *SLB* which increases with the number of bin-types. Results for $m = 5$ and $n = 200$,$n = 500$ are not shown because the time limit was attained in the calculation of the optimal solution, but this limit wasn't attained in the *CLB* and *SLB* solution calculation.

Results of comparison between *SLB* and *CLB* are obviously awaited, insofar as contrary to *CLB*, the *SLB* takes into account only the lowest bin-types unit cost with infinite capacity. Then, optimal solutions are given when the sum of items size is divisible by the capacity of the taken bin-type. This is why optimal solutions are given for only two instances over 240 we have tested. Finally, from these results it should be noted that overall, with *CLB* we were able to find an adequate continuous lower bound for the VSBPP in a lower computing time and consequently the *CLB* can substitute the optimal solution to evaluate any resolution methods.

5 Conclusion

The **VSBPP** is a hard combinatorial optimization problem of practical interest. It is a generalization of the well-known one dimensional BPP where we consider more than one category of bins. In this paper we have discussed a particular case of the VSBPP where sizes and costs of bin-types satisfy monotonicity constraints (Kang & Park 2003). For this problem, we have described a new continuous lower bound based on generalization of the well known continuous lower bound for the BPP; used as a basis to compare different algorithms. Numerical experiments on VSBPP-instances show the relevance of our continuous lower bound and its relatively low computing time, in particular when the quotient between both costs of bin-types is important and when items are small than possible.

REFERENCES

Coffmann, J. E. G., Garey, M. R. & Johnson, D. S. (1997). *Approximation algorithms for NP-hard problems: Approximation algorithms for bin-packing-a survey*, PWS Publishing, Boston, pp. 46–93.

Correia, I., Gouveia, L. & da Gama, F. S. (2008). Solving the variable size bin-packing problem with descritized formulations, *Computers & Operations Research* **35**: 2103–2113.

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

Epstein, L. & Favrholdt, L. M. (2002). On-line maximizing the number of items packed in variable-sized bins, *in* O. H. Ibarra & L. Zhang (eds), *Lecture Notes in Computer Science 2387*, COCOON 2002, Springer-Verlag, Berlin Heidelberg, pp. 467–475.

Friesen, D. K. & Langston, M. A. (1986). Variable sized bin-packing, *SIAM Journal on Computing* **15**: 222–230.

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intratability: a guide to the theory of NP-completeness*, Freeman, San Francisco.

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

Martello, S. & Toth, P. (1990b). Lower bounds and reduction procedures for the bin packing problem, *Discrete Applied Mathematics* **28**: 59–70.

Monaci, M. (2002). *Algorithms for packing and scheduling problems*, Phd thesis or/02/4, Università di Bologna.

Murgolo, F. D. (1987). An efficient approximation scheme for variable-sized bin-packing, *SIAM Journal on Computing* **16**(1): 149–161.

Righini, G., Bettinelli, A. & Ceselli, A. (2010). A branch-and-price algorithm for the variable size bin-packing problem with minimum filling constraint, *Annals of Operations Research* **179**(1): 221–241.