Bin Packing with Linear Usage Costs - An Application to Energy Management in Data Centres

Hadrien Cambazard, Deepak Mehta, Barry O'Sullivan, Helmut Simonis

Bin Packing with Linear Usage Costs – An Application to Energy Management in Data Centres*

Hadrien Cambazard[1], Deepak Mehta[2], Barry O'Sullivan[2], and Helmut Simonis[2]

[1] G-SCOP, Université de Grenoble; Grenoble INP; UJF Grenoble 1; CNRS, France
hadrien.cambazard@grenoble-inp.fr
[2] Cork Constraint Computation Centre, University College Cork, Ireland
{d.mehta|b.osullivan|h.simonis}@4c.ucc.ie

Abstract. EnergeTIC is a recent industrial research project carried out in Grenoble on optimizing energy consumption in data-centres. The efficient management of a data-centre involves minimizing energy costs while ensuring service quality. We study the problem formulation proposed by EnergeTIC. First, we focus on a key sub-problem: a bin packing problem with linear costs associated with the use of bins. We study lower bounds based on Linear Programming and extend the bin packing global constraint with cost information. Second, we present a column generation model for computing the lower bound on the original energy management problem where the pricing problem is essentially a cost-aware bin packing with side constraints. Third, we show that the industrial benchmark provided so far can be solved to near optimality using a large neighborhood search.

1 Introduction

Energy consumption is one of the most important sources of expense in data centers. The ongoing increase in energy prices (a 50% increase is forecasted by the French senate by 2020) and the growing market for cloud computing are the main incentives for the design of energy efficient centers. We study a problem associated with the EnergeTIC[3] project which was accredited by the French government (FUI) [2]. The objective is to control the energy consumption of a data center and ensure that it is consistent with application needs, economic constraints and service level agreements. We focus on how to reduce energy cost by taking variable CPU requirements of client applications, IT equipment and virtualization techniques into account.

There are a variety of approaches to energy management in data centres, the most well-studied of which is energy-aware workload consolidation. A Mixed Integer Programming (MIP) approach to dynamically configuring the consolidation of multiple services or applications in a virtualised server cluster has been proposed in [16]. That work focused on power efficiency and considered the costs of turning on/off the servers.

* The authors acknowledge their industrial partners (Bull, Schneider Electric, Business & Decision and UXP) as well as several public research institutions (G2Elab, G-SCOP and LIG). The authors from UCC are supported by Science Foundation Ireland Grant No. 10/IN.1/I3032.

[3] Minalogic EnergeTIC is a Global competitive cluster located in Grenoble France and fostering research-led innovation in intelligent miniaturized products and solutions for industry.

However, workloads were entirely homogeneous and there was little uncertainty around the duration of tasks. Constraint Programming is used in [8] with a different cost model.

A combinatorial optimisation model for the problem of loading servers to a desired utilisation level has, at its core, a bin packing (BP) problem [20]. In such a model each server is represented by a bin with a capacity equal to the amount of resource available. Bin packing is a very well studied NP-Hard problem. A significant amount of work has been conducted on lower bounds [13], approximation and exact algorithms. Although this research is still very active as demonstrated by the recent progress [17], researchers have started to look at variants involved in industrial applications.

In Section 2 we present an extension of bin packing which is a key sub-problem of the application domain and we show how to handle it efficiently with Constraint Programming (CP). In Section 3 we study the formulation of the EnergeTIC problem. In particular a lower bound computation technique is designed to assert the quality of the upper bounds found by a large neighborhood search. Section 4 reports the experiments on a real data-set followed by conclusions in Section 5.

2 Bin Packing with Linear Usage Costs

We consider a variant of the Bin Packing problem (BP) [20], which is the key sub-problem of the application investigated here. We denote by $S = \{w_1, \ldots, w_n\}$ the integer sizes of the n items such that $w_1 \leq w_2 \leq \ldots w_n$. A bin j is characterized by an integer capacity C_j, a non-negative fixed cost f_j and a non-negative cost c_j for each unit of used capacity. We denote by $B = \{\{C_1, f_1, c_1\}, \ldots, \{C_m, f_m, c_m\}\}$ the characteristics of the m bins. A bin is used when it contains at least one item. Its cost is a linear function $f_j + c_j l_j$, where l_j is the total size of the items in bin j. The total load is denoted by $W = \sum_{i=1}^{n} w_i$ and the maximum capacity by $C_{max} = max_{1 \leq j \leq m} C_j$. The problem is to assign each item to a bin subject to the capacity constraints so that we minimize the sum of the costs of all bins. We refer to this problem as the *Bin Packing with Usage Cost* problem (BPUC). BP is a special case of BPUC where all f_j are set to 1 and all c_j to 0. The following example shows that a good solution for BP might not yield a good solution for BPUC.

Example 1. In Figure 1, Scenario 1, B ={(9,0,1),(3,0,2),(3,0,2),(3,0,2),(3,0,2)} and S = {2,2,2,2,3,3,3}. Notice that $\forall j, f_j = 0$. The packing (P_1) : {{2,2,2,2}, {3}, {3}, {3}, {}} is using the minimum number of bins and has a cost of 26 (8*1 + 3*2 + 3*2 + 3*2). The packing (P_2): {{3,3,3}, {2}, {2}, {2}, {2}} is using one more bin but has a cost of 25 (9 + 2*2 + 2*2 + 2*2 + 2*2). Here, (P_2) is better than (P_1) and using the minimum number of bins is not a good strategy. Now change the last unit cost to $c_5 = 3$ (see Figure 1, Scenario 2). The cost of (P_1) remains unchanged since it does not use bin number 5 but the cost of (P_2) increases to 27, and thus (P_1) is now better than (P_2).

Literature Review. A first relevant extension of BP for the current paper is called Variable Size Bin-Packing, where bins have different capacities and the problem is to minimize the sum of the wasted space over all used bins [15]. It can be seen as a special case of BPUC where all $f_j = C_j$ and $c_j = 0$. Recent lower bounds and an exact approach are examined in [11]. A generalization to any kind of fixed cost is presented in [5], which can be seen as a special case of BPUC where all $c_j = 0$. Concave costs of bin utilization

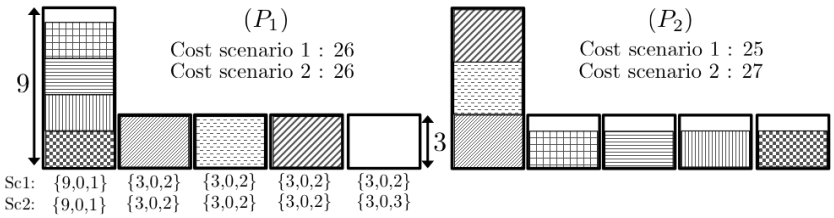


Fig. 1. Example of optimal solutions in two scenarios of costs. In Scenario 1, the best solution has no waste on the cheapest bin. In Scenario 2, it does not fill completely the cheapest bin.

studied in [12] are more general than the linear cost functions of BPUC. However [12] does not consider bins of different capacities and deals with the performance of classical BP heuristics whereas we are focusing on lower bounds and exact algorithms. Secondly, BP with general cost structures have been introduced [3] and studied [9]. The authors investigated BP with non-decreasing and concave cost functions of the number of items put in a bin. They extend it with profitable optional items in [4]. Their framework can model a fixed cost but does not relate to bin usage.

2.1 Basic Formulation and Lower Bounds

Numerous linear programming models have been proposed for BP [7]. We first present a formulation for BPUC. For each bin a binary variable y_j is set to 1 if bin j is used in the packing, and 0 otherwise. For each item $i \in \{1,\dots,n\}$ and each bin $j \in \{1,\dots,m\}$ a binary variable x_{ij} is set to 1 if item i is packed into bin j, and 0 otherwise. We add non-negative variables l_j representing the load of each bin j. The model is as follows:

$$
\begin{array}{lll}
\textbf{Minimize} & z_1 = \sum_{j=1}^{m}(f_j y_j + c_j l_j) & \\
(1.1) & \sum_{j=1}^{m} x_{ij} = 1, & \forall i \in \{1,\dots,n\} \\
(1.2) & \sum_{i=1}^{n} w_i x_{ij} = l_j, & \forall j \in \{1,\dots,m\} \\
(1.3) & l_j \le C_j y_j, & \forall j \in \{1,\dots,m\} \\
(1.4) & x_{ij} \in \{0,1\}, y_j \in \{0,1\}, l_j \ge 0 & \forall j \in \{1,\dots,m\}, \forall i \in \{1,\dots,n\}
\end{array}
\tag{1}
$$

Constraint (1.1) states that each item is assigned to one bin whereas (1.2) and (1.3) enforce the capacity of the bins. We now characterize the linear relaxation of the model. Let $r_j = f_j/C_j + c_j$ be a real number associated with bin j. If bin j is filled completely, r_j is the cost of one unit of space in bin j. We sort the bins by non-decreasing r_j: $r_{a_1} \le r_{a_2} \le \ldots \le r_{a_m}$; $a_1,\dots,a_m$ is a permutation of the bin indices $1,\dots,m$. Let k be the minimum number of bins such that $\sum_{j=1}^{k} C_{a_j} \ge W$.

Proposition 1. *Let z_1^* be the optimal value of the linear relaxation of the formulation (1). We have $z_1^* \ge Lb_1$ with $Lb_1 = \sum_{j=1}^{k-1} C_{a_j} r_{a_j} + (W - \sum_{j=1}^{k-1} C_{a_j}) r_{a_k}$.*

Proof. $z_1^* = \sum_{j=1}^{m}(f_j y_j + c_j l_j) \ge \sum_{j=1}^{m}(f_j \frac{l_j}{C_j} + c_j l_j)$ because of the constraint $l_j \le C_j y_j$, so $z_1^* \ge \sum_{j=1}^{m}(\frac{f_j}{C_j} + c_j) l_j \ge \sum_{j=1}^{m} r_j l_j$. Lb_1 is the quantity minimizing $\sum_{j=1}^{m} r_j l_j$ under the constraints $\sum_j l_j = W$ where each $l_j \le C_j$. To minimize the quantity we must split W over the l_j related to the smallest coefficients r_j. Hence, $z_1^* \ge \sum_{j=1}^{m} r_j l_j \ge Lb_1$. □

Lb_1 is a lower bound of BPUC that can be easily computed. Also notice that Lb_1 is the bound that we get by solving the linear relaxation of formulation (1).

Proposition 2. *Lb_1 is the optimal value of the linear relaxation of the formulation (1).*

Proof. For all $j < k$, we set each y_{a_j} to 1 and l_{a_j} to C_j. We fix l_{a_k} to $(W - \sum_{j=1}^{k-1} C_{a_j})$ and y_{a_k} to l_{a_k}/C_{a_k}. For all $j > k$ we set $y_{a_j} = 0$ and $l_{a_j} = 0$. Constraints (1.3) are thus satisfied. Finally we fix $x_{i,a_j} = \frac{l_{a_j}}{W}$ for all i, j so that constraints (1.2) and (1.1) are satisfied. This is a feasible solution of the linear relaxation of (1) achieving an objective value of Lb_1. We have, therefore, $Lb_1 \geq z_1^*$ and consequently $z_1^* = Lb_1$ from Proposition 1. □

Adding the constraint $x_{ij} \leq y_j$ for each item i and bin j, strengthens the linear relaxation only if $W < C_{a_k}$. Indeed, the solution given in the proof is otherwise feasible for the constraint, ($\forall j < k,\ x_{i,a_j} = \frac{l_{a_j}}{W} \leq y_{a_j} = 1$ and for $j = k$ we have $\frac{l_{a_k}}{W} \leq \frac{l_{a_k}}{C_{a_k}}$ if $W \geq C_{a_k}$).

2.2 Two Extended Formulations of BPUC

The Cutting Stock Model. The formulation of Gilmore and Gomory for the cutting stock problem [10] can be adapted for BPUC. The items of equal size are now grouped and for $n' \leq n$ different sizes we denote the number of items of sizes $w'_1, \ldots, w'_{n'}$ by $q_1, \ldots, q_{n'}$ respectively. A cutting pattern for bin j is a combination of item sizes that fits into bin j using no more than q_d items of size w'_d. In the i-th pattern of bin j, the number of items of size w'_d that are in the pattern is denoted g_{dij}. Let I_j be the set of all patterns for bin j. The cost of the i-th pattern of bin j is therefore equal to $co_{ij} = f_j + (\sum_{d=1}^{n'} g_{dij} w'_d) c_j$. The cutting stock formulation is using a variable p_{ij} for the i-th pattern of bin j:

$$
\begin{array}{lll}
\textbf{Minimize} & z_2 = \sum_{j=1}^{m} \sum_{i \in I_j} co_{ij} p_{ij} & \\
(2.1) & \sum_{j=1}^{m} \sum_{i \in I_j} g_{dij} p_{ij} = q_d & \forall d \in \{1, \ldots, n'\} \\
(2.2) & \sum_{i \in I_j} p_{ij} = 1 & \forall j \in \{1, \ldots, m\} \\
(2.3) & p_{ij} \in \{0, 1\} & \forall j \in \{1, \ldots, m\}, i \in I_j
\end{array}
\tag{2}
$$

Constraint (2.1) states that each item has to appear in a pattern (thus in a bin) and (2.2) enforces one pattern to be designed for each bin (convexity constraints). A pattern p_{ij} for bin j is valid if $\sum_{d=1}^{n'} g_{dij} w'_d \leq C_j$ and all g_{dij} are integers such that $q_d \geq g_{dij} \geq 0$. The sets I_j have an exponential size so the linear relaxation of this model can be solved using column generation. The pricing step is a knapsack problem that can be solved efficiently by dynamic programming if the capacities are small enough.

The* Arc-Flow *Model. Carvalho introduced an elegant Arc-Flow model for BP [6,7]. His model explicitly uses each unit of capacity of the bins. In the following we show how to adapt it for BPUC. Consider a multi-graph $G(V, A)$, where $V = \{0, 1, ..., C_{max}\} \cup \{F\}$ is the set of $C_{max} + 2$ nodes labeled from 0 to C_{max} and a final node labeled F, and $A = I \cup J$ is the set of two kinds of edges. An edge $(a, b) \in I$ between two nodes labelled $a \leq C_{max}$ and $b \leq C_{max}$ represents the use of an item of size $b - a$. An edge of $(a, F) \in J$ for each bin j represents the usage a of the bin j, and therefore $a \leq C_j$. An example of

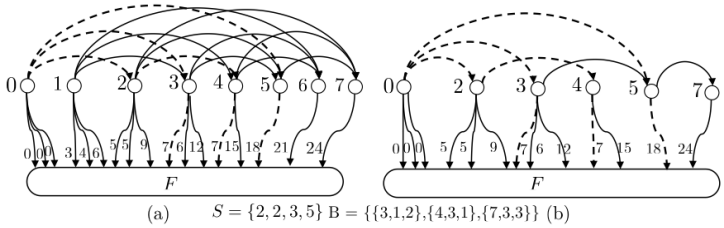


Fig. 2. (a) An example of the graph underlying the Arc-Flow model for $S = \{2, 2, 3, 5\}$, $B = \{\{3, 1, 2\}, \{4, 3, 1\}, \{7, 3, 3\}\}$ so that $C_{max} = 7$. A packing is shown using a dotted line: $\{3\}$ is put in the first bin for a cost of 7, $\{2, 2\}$ is in the second bin for a cost of 7 and $\{5\}$ in the last bin for a cost of 18. (b) The graph underlying the Arc-Flow model after the elimination of symmetries.

such a graph is shown in Figure 2(a). Notice that this formulation has symmetries since a packing can be encoded by many different paths. Some reduction rules were given by Carvalho [6], which help in reducing such symmetries (see Figure 2(b)).

BPUC can be seen as a minimum cost flow between 0 and F with constraints enforcing the number of edges of a given length used by the flow to be equal to the number of items of the corresponding size. We have variables x_{ab} for each edge $(a, b) \in I$ as well as variables y_{aj} for each pair of bin $j \in \{1, \ldots, m\}$ and $a \in V$. The cost of using an edge $(a, F) \in J$ for bin j with $a > 0$ is $co_{aj} = f_j + a \cdot c_j$ and $co_{0j} = 0$. The model is as follows:

$$
\begin{array}{lll}
\textbf{Minimize} & z_3 = \sum_{j=1}^{m} \sum_{k=0}^{k=C_{max}} co_{kj} y_{kj} & \\
(3.1) & \sum_{(a,b)\in A} x_{ab} - \sum_{(b,c)\in A} x_{bc} - \sum_{j=1}^{m} y_{bj} = \begin{cases} 0 & \forall b \in \{1, 2, \ldots, C_{max}\} \\ -m \text{ for } b = 0 \end{cases} & \\
(3.2) & \sum_{a=0}^{C_j} y_{aj} = 1 & \forall j \in \{1, \ldots, m\} \\
(3.3) & \sum_{(k,k+w'_d)\in A} x_{k,k+w'_d} = q_d & \forall d \in \{1, 2, \ldots, n'\} \\
(3.4) & y_{aj} = 0 & \forall (j, a) \in \{1, \ldots, m\} \times \{C_j + 1, \ldots, C_{max}\} \\
(3.5) & x_{ab} \in \mathbb{N} & \forall (a, b) \in A \\
(3.6) & y_{aj} \in \{0, 1\} & \forall (j, a) \in \{1, \ldots, m\} \times \{0, \ldots, C_{max}\}
\end{array}
\tag{3}
$$

Constraint (3.1) enforces the flow conservation at each node, and Constraint (3.2) states that each bin should be used exactly once. Constraint (3.3) ensures that all the items are packed, while Constraint (3.4) enforces that bin j is not used beyond its capacity C_j. A solution can be obtained again by decomposing the flow into paths. The number of variables in this model is in $O((n' + m) \cdot C_{max})$ and the number of constraints is $O(C_{max} + m + n')$. Although its LP relaxation is stronger than that of Model (1), it remains dominated by that of Model (2).

Proposition 3. $z_3^* \leq z_2^*$. *The optimal value of the linear relaxation of (3) is less than the optimal value of the linear relaxation of (2).*

Proof. Let (p^*) be a solution of the linear relaxation of (2). Each pattern p_{ij}^* is mapped to a path of the Arc-Flow model. A fractional value p_{ij}^* is added on the arcs corresponding to the item sizes of the pattern (the value of the empty patterns for which all $g_{dij} = 0$ is put on the arcs y_{0j}). The flow conservation (3.1) is satisfied by construction, so is (3.2) because of (2.2) and so are the demand constraints (3.3) because of (2.1). Any solution of (2) is thus encoded as a solution of (3) for the same cost so $z_3^* \leq z_2^*$. □

Proposition 4. *z_2^* can be stronger than z_3^** i.e *there exist instances such that $z_2^* > z_3^*$.*

Proof. Consider the following instance: $S = \{1, 1, 2\}$ and $B = \{\{3, 1, 1\}, \{3, 4, 4\}\}$. Two items of size 1 occurs so that $n^{'} = 2$, $q_1 = 2, q_2 = 1$ corresponding to $w_1^{'} = 1, w_2^{'} = 2$. The two bins have to be used and the first *dominates* the second (the maximum possible space is used in bin 1 in any optimal solution) so the optimal solution is the packing $\{\{2, 1\}, \{1\}\}$ (cost of 12). Let's compute the value of z_2^*. It must fill the first bin with the pattern $[g_{111}, g_{211}] = [1, 1]$ for a cost of 4. Only three possible patterns can be used to fill the second bin: $[0, 0]$, $[1, 0]$ and $[2, 0]$ (a valid pattern p_{i2} is such that $g_{1i2} \leq 2$). The best solution is using $[g_{112}, g_{212}] = [2, 0]$ and $[g_{122}, g_{222}] = [0, 0]$ taking both a 0.5 value to get a total cost $z_2^* = 4 + 6 = 10$. The Arc-Flow model uses a path to encode the same first pattern [1,1] for bin 1. But it can build a path for bin 2 with a $\frac{1}{3}$ unit of flow taking three consecutive arcs of size 1 to reach a better cost of $\frac{1}{3} * 16 \approx 5,33$. This path would be a pattern [3,0] which is not valid for (2). So $z_3^* \approx 9.33$ and $z_2^* > z_3^*$. □

The Arc-Flow model may use a path containing more than q_d arcs of size $w_d^{'}$ with a positive flow whereas no such patterns exist in (3) **because** the sub-problem is subject to the constraint $0 \leq g_{dij} \leq q_d$. The cutting stock formulation used in [6] ignores this constraint and therefore the bounds are claimed to be equivalent.

2.3 Extending the Bin Packing Global Constraint

A bin packing global constraint was introduced in CP by [19]. We present an extension of this global constraint to handle BPUC. The scope and parameters are as follows:

$$\text{BinPackingUsageCost}([x_1, \ldots, x_n], [l_1, \ldots, l_m], [y_1, \ldots, y_m], b, z, S, B)$$

Variables $x_i \in \{1, \ldots, m\}$, $l_j \in [0, \ldots, C_j]$ and $b \in \{1, \ldots, m\}$ denote the bin assigned to item i, the load of bin j, and the number of bins used, respectively. These variables are also used by the BinPacking constraint. Variables $y_i \in \{0, 1\}$ and $z \in \mathbb{R}$ are due to the cost. They denote whether bin j is open, and the cost of the packing. The last two arguments refers to BPUC and give the size of the items as well as the costs (fixed and unit). In the following, $\underline{x}$ (resp. $\overline{x}$) denotes the lower (resp. upper) bound of variable x.

Cost-based Propagation using Lb_1. The characteristics of the bins of the restricted BPUC problem based on the current state of the domains of the variables is denoted by $B^{'}$, and defined by $B^{'} = \{\{C_1^{'}, f_1^{'}, c_1\}, \ldots, \{C_m^{'}, f_m^{'}, c_m\}\}$ where $C_j^{'} = \overline{l_j} - \underline{l_j}$, and $f_j^{'} = (1 - \underline{y_j}) f_j$. The total load that remains to be allocated to the bins is denoted $\overline{W^{'}} = W - \sum_{j=1}^{m} \underline{l_j}$. Notice that we use the lower bounds of the loads rather than the already packed items. We assume it is strictly better due to the reasonings of the bin packing constraint.

Lower bound of z. The first propagation rule is the update of the lower bound $\underline{z}$ of z. The bound is summing the cost due to open bins and minimum loads with the value of Lb_1 on the remaining problem. It gives a maximum possible cost increase gap:

$$Lb_1^{'} = \sum_{j=1}^{m} (\underline{l_j} c_j + \underline{y_j} f_j) + Lb_1(W^{'}, B^{'}); \qquad \underline{z} \leftarrow max(\underline{z}, Lb_1^{'}); \qquad gap = \overline{z} - Lb_1^{'} \quad (4)$$

Bounds of the load variables. We define the bin packing problem B'' obtained by excluding the space supporting the lower bound $Lb_1(W', B')$. Lb_1 is using L_j units of space on bin a_j. The bins $a_1, \ldots, a_{k-1}$ are fully used so $\forall j < k, L_j = C'_{a_j}$, for bin a_k we have $L_k = W' - \sum_{j=1}^{k-1} C'_{a_j}$ and $\forall j > k, L_j = 0$. The resulting bins are defined as $B'' = \{\{C''_1, f'_1, c_1\}, \ldots, \{C''_m, f'_m, c_m\}\}$ where $C''_{a_j} = 0$ for all $j < k$, $C''_{a_k} = C'_{a_k} - L_k$ and $C''_{a_j} = C'_{a_j}$ for all $j > k$. Lower and upper bounds of loads are adjusted with rules (5).

Let $q^1_{a_j}$ be the largest quantity that can be removed from a bin a_j, with $j \leq k$, and put at the other cheapest possible place without overloading $\overline{z}$. Consequently, when $j < k$, $q^1_{a_j}$ is the largest value in $[1, L_j]$ such that $(Lb_1(q^1_{a_j}, B'') - q^1_{a_j} r_{a_j}) \leq gap$. When $j = k$, the same reasonning can be done by setting $C''_{a_k} = 0$ in B''.

Similarly, let $q^2_{a_j}$ be the largest value in $[1, C'_{a_j}]$ that can be put on a bin a_j, with $j \geq k$, without triggering a contradiction with the remaining gap of cost. $q^2_{a_j}$ is thus the largest value in $[1, C'_{a_j}]$ such that $(q^2_{a_j} r_{a_j} - (Lb_1(W', B') - Lb_1(W' - q^2_{a_j}, B'))) \leq gap$.

$$\forall j \leq k, \quad \underline{l_{a_j}} \leftarrow \underline{l_{a_j}} + L_j - q^1_{a_j}; \qquad\qquad \forall j \geq k, \quad \overline{l_{a_j}} \leftarrow \underline{l_{a_j}} + q^2_{a_j}. \quad (5)$$

Channeling. The constraint ensures two simple rules relating the load and open-close variables (a bin of zero load can be open): $y_j = 0 \implies l_j = 0$ and $l_j > 0 \implies y_j = 1$.
Bounds of the open-close variables. The propagation rule for $\underline{l_j}$ can derive $\underline{l_j} > 0$ from (5), which in turn (because of the channeling between y and l) will force a bin to open *i.e* $y_{a_j} \in \{0, 1\}$ will change to $y_{a_j} = 1$. To derive that a y_j has to be fixed to 0, we can use Lb_1 similarly to the reasonings presented for the load variables (checking that the increase of cost for opening a bin remains within the gap).

Tightening the bounds of the load variables can trigger the existing filtering rules of the bin packing global constraint thus forbidding or committing items to bins. Notice that items are only increasing the cost indirectly by increasing the loads of the bins because the cost model is defined by the state of the bins (rather than the items). The cost-based propagation on x is thus performed by the bin packing global constraint solely as a consequence of the updates on the bin related variables, i.e. l and y.

Algorithms and Complexity. Assuming that B' and W' are available, $Lb_1(W', B')$ can be computed in $O(m\,log(m))$ time. Firstly we compute the r_j values corresponding to B' for all bins. Secondly, we sort the bins in non-decreasing r_j. Finally, the bound is computed by iterating over the sorted bins and the complexity is dominated by the sorting step. After computing $Lb_1(W', B')$, the values a_j (the permutation of the bins) such that $r_{a_1} \leq r_{a_2} \leq \ldots \leq r_{a_m}$ are available as well as the critical k and $L_k = W' - \sum_{j=1}^{k-1} C'$. The propagation of $\underline{l_{a_j}}$ and $\overline{l_{a_j}}$ can then be done in $O(m)$ as shown in Figure 3.

3 Application – Energy Optimization in a Data Centre

The system developed by EnergeTIC is based on a model of the energy consumption of the various components in a data centre, a prediction system to forecast the demand and an optimization component computing the placement of virtual machines onto servers.

Algorithm 1: UpdateMinimumLoad
Input: a_j with $j \leq k$, B', gap
Output: a lower bound of l_{a_j}

1. $costInc = 0$; $q = 0$; $b = k$;
2. **If** $(j == k)$ $\{b = k + 1;\}$
3. **While** $(q < L_j$ && $b \leq m)$
4. $\quad loadAdd = \min(L_j - q, C'_{a_b} - L_b)$;
5. $\quad costIncb = loadAdd \times (r_{a_b} - r_{a_j})$;
6. $\quad$ **If** $((costIncb + costInc) > gap)$
7. $\quad\quad q = q + \lfloor \frac{gap - costInc}{r_{a_b} - r_{a_j}} \rfloor$;
8. $\quad\quad$ **return** $L_j + \underline{l_{a_j}} - q$;
9. $\quad costInc = costInc + costIncb$;
10. $\quad q = q + loadAdd$; $b = b + 1$;
11. **return** $\underline{l_{a_j}}$

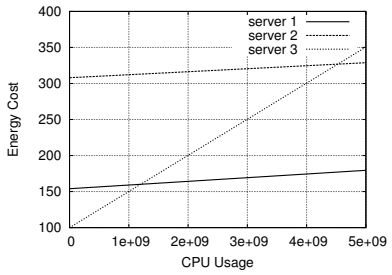
Algorithm 2: UpdateMaximumLoad
Input: a_j with $j \geq k$, B', gap
Output: an upper bound of l_{a_j}

1. $costInc = 0$; $q = 0$; $b = k$;
2. **If** $(j == k)$ $\{q = L_k$; $b = k - 1;\}$
3. **While** $(q < C'_{a_j}$ && $b \geq 0)$
4. $\quad loadAdd = min(L_b, C'_{a_j} - q)$;
5. $\quad costIncb = loadAdd \times (r_{a_j} - r_{a_b})$;
6. $\quad$ **If** $((costIncb + costInc) > gap)$
7. $\quad\quad q = q + \lfloor \frac{gap - costInc}{r_{a_j} - r_{a_b}} \rfloor$;
8. $\quad\quad$ **return** $\overline{l_{a_j}} + q$;
9. $\quad costInc = costInc + costIncb$;
10. $\quad q = q + loadAdd$; $b = b - 1$;
11. **return** $\overline{l_{a_j}}$

Fig. 3. Propagation algorithms for updating the lower and upper bounds of the load variables

Energy Model. In the last decade green data centres have focused on limiting the amount of energy that is not used for running the client's applications. The Power Usage Effectiveness (PUE) is a key indicator introduced by the Green Grid consortium [1] which measures the ratio between the total energy consumption of the data centre and the energy used by its IT systems (e.g., servers, networks, etc.). A value of 1 is the perfect score. The current average in industry is around 1.7 and the most efficient data centres are reaching 1.4 or even 1.2. As not all electrical power consumed by the IT equipment is transformed into a useful work product, the need to refine such a metric arose quickly. Therefore, the Green Grid proposed a very fine grained indicator for that purpose [1]. This metric, although very accurate, is not really used in practice because of its complexity and no consensus has been reached for a practical and relevant indicator. The EnergeTIC project introduced a new energy indicator which is defined as the ratio between the total energy consumed and the energy really used to run clients' applications. This indicator however relies on a model of the energy consumption of each equipment. A system, based on three different servers (quad-, bi- and mono- processor) with different energy behaviors, was provided by Bull to perform the measurements. As an example, the energy cost of the power consumption of three different servers at different CPU loads taken from one of the problem instances is shown in Figure 4.

Demand Model. The demands of the real benchmarks used in the experimental section are coming from the Green Data Centre of Business & Decision Eolas located in Grenoble. It was used to study and validate the system operationally. It is instrumented with thousands of sensors spread over the site to monitor the energy consumption of the centre and claims a PUE between 1.28 and 1.34. It deals with an heterogeneous demand: web applications, e-commerce, e-business, e-administrations. An example showing variable requirements of CPU usage over 24 time-periods for multiple virtual machines taken from one of the problem instances is shown in Figure 5.



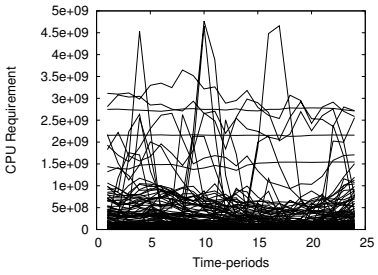


Fig. 4. Energy cost vs CPU Usage for 3 servers. **Fig. 5.** Variable demands of virtual machines.

3.1 Problem Description and Notation

The problem is to place a set of virtual machines on a set of servers over multiple time-periods to minimize the energy cost of the data center. The CPU usage of a VM is changing over time. At each period, we must ensure that the virtual machines have enough resources (CPU and memory). Let $VM = \{v_1, \ldots, v_n\}$ be the set of virtual machines, $SE = \{s_1, \ldots, s_m\}$ be the set of servers and $T = \{p_1, \ldots, p_h\}$ be the set of periods.

Virtual Machines. A virtual machine v_i is characterized by a memory consumption M_i independent of the time-period, a set $SA_i \subseteq S$ of allowed servers where it can be hosted, and a potential initial server (for time-period p_0) denoted by $Iserv_i$ (which might be unknown). A virtual machine v_i has a CPU consumption U_{it} at time-period t.

Servers. A server s_j can be in two different states: ON=1 or STBY=0 (stand-by). It is characterized by: a CPU capacity $Umax_j$; a memory capacity $Mmax_j$; a fixed cost of usage $Emin_j$ (in Watts); a unit cost τ_j per unit of used capacity; a basic CPU consumption Ca_j when it is ON (to run the operating system and other permanent tasks); an energy consumption $Esby_j$ when it is in state STBY; an energy consumption $Esta_j$ to change the state of the server from STBY to ON; an energy consumption $Esto_j$ to change the state of the server from ON to STBY; a maximum number $Nmax_j$ of virtual machines that can be allocated to it at any time period; a set of periods $P_j \subseteq T$ during which s_j is forced to be ON; and a potential initial state $Istate_j \in \{0, 1\}$.

If a server is ON, its minimum cost is $Emin_j + \tau_j Ca_j$, and if it is STBY, its cost is $Estby_j$. For the sake of simplicity, to compute the fixed energy cost of an active server we include the basic consumption Ca_j and the standby energy $Estby_j$ in $Emin_j$ so that $Emin'_j = Emin_j - Estby_j + \tau_j Ca_j$. This way we can state the BinPackingUsageCost directly with the semantic given earlier by adding the constant $\sum_{s_j \in SE} Estby_j$ in the final objective value. We also shift the CPU capacity of the servers: $Umax'_j = Umax_j - Ca_j$.

Migrations. The maximum number of changes of servers among all virtual machines from one period to the next is denoted by N and the cost of a migration by $Cmig$.

The problem can be seen as a series of cost-aware bin packing problems (one per period) in two dimensions (CPU and memory) that are coupled by the migration constraints and the cost for changing the state of a server. Figure 6 gives an overview of the problem. This example has four servers, each shown by a rectangle whose dimensions are representing the CPU and memory capacities. A virtual machine is a small rectangle whose height (its CPU) varies from one period to the next. In this scenario, the CPU needs

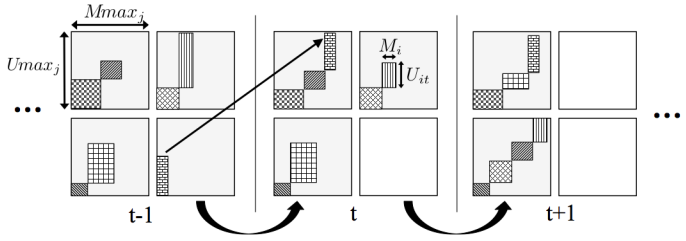


Fig. 6. A solution over three time periods. Virtual machines migrate to turn off two servers at $t+1$.

$$\textbf{Minimize} \ \sum_{s_j \in SE} \sum_{t \in T} (Esta_j bto_{jt} + Esto_j otb_{jt} + \tau_j cpu_{jt} + Emin'_j o_{jt}) + Cmig(\sum_{v_i \in VM} \sum_{t \in T} a_{it})$$

$$\begin{array}{lllll}
(6.1) & \sum_{s_j \in SE} x_{ijt} & = 1 & (\forall\, v_i \in VM, p_t \in T) & \\
(6.2) & x_{ijt} & = 0 & (\forall\, v_i \in VM, p_t \in T, s_j \notin SA_i) & \\
(6.3) & x_{ijt} & \leq o_{jt} & (\forall\, v_i \in VM, p_t \in T, s_j \in SE) & \\
(6.4) & cpu_{jt} & = \sum_{v_i \in VM} U_{it} x_{ijt} & (\forall\, s_j \in SE, p_t \in T) & \\
(6.5) & cpu_{jt} & \leq Umax'_j o_{jt} & (\forall\, s_j \in SE, p_t \in T) & \\
(6.6) & \sum_{v_i \in VM} M_i x_{ijt} & \leq Mmax_j o_{jt} & (\forall\, s_j \in SE, p_t \in T) & \\
(6.7) & \sum_{v_i \in VM} x_{ijt} & \leq Nmax_j o_{jt} & (\forall\, s_j \in SE, p_t \in T) & \\
(6.8) & mig_{ijt} & \geq x_{ijt} - x_{ijt-1} & (\forall\, v_i \in VM, s_j \in SE, p_t \in T) & (6) \\
(6.8') & a_{it} & \geq \sum_{s_j \in SE} mig_{ijt} & (\forall\, v_i \in VM, p_t \in T) & \\
(6.9) & \sum_{v_i \in VM} a_{it} & \leq N & (\forall\, p_t \in T) & \\
(6.10) & bto_{jt} & \geq o_{jt} - o_{jt-1} & (\forall\, s_j \in SE, p_t \in T) & \\
(6.11) & otb_{jt} & \geq o_{jt-1} - o_{jt} & (\forall\, s_j \in SE, p_t \in T) & \\
(6.12) & o_{jt} & = 1 & (\forall\, s_j \in SE, p_t \in P_j) & \\
(6.13) & x_{ij0} & = 0 & (\forall\, v_i \in VM, s_j \in SE - \{Iserv_i\}) & \\
(6.14) & x_{i,Iserv_i,0} & = 1 & (\forall\, v_i \in VM) & \\
(6.15) & o_{j0} & = Istate_j & (\forall\, s_j \in SE) &
\end{array}$$

of some virtual machines decrease allowing to find better packings and turn off two servers at $t + 1$.

3.2 An Integer Linear Model

We present the integer linear model of the problem in which the following variables are used: $x_{ijt} \in \{0, 1\}$ indicates whether virtual machine v_i is placed on server s_j at time t. $cpu_{jt} \in [0, Umax'_j]$ gives the CPU consumption of s_j at period t. $o_{jt} \in \{0, 1\}$ is set to 1 if s_j is ON at time t, 0 otherwise. $bto_{jt} \in \{0, 1\}$ is set to 1 if s_j was in STBY at $t - 1$ and is turned ON at t. $otb_{jt} \in \{0, 1\}$ is set to 1 if s_j was in ON at $t - 1$ and is put STBY at t. $mig_{ijt} \in \{0, 1\}$ is set to 1 if v_i is on s_j at time t and was on a different server at $t - 1$. $a_{it} \in \{0, 1\}$ is set to 1 if v_i is on a different server at t than the one it was using at $t - 1$.

The initial state is denoted by $t = 0$. The model is summarized in Model (6). We omit the constant term $\sum_{s_j \in SE} Estby_j$ from the objective function. Constraint (6.1) states that a virtual machine has to be on a server at any time; (6.2) enforces the forbidden servers for each machine; (6.3) enforces a server to be ON if it is hosting at least one

virtual machine; (6.4) links the CPU load of a server to the machines assigned to it. (6.5–6.7) are the resource constraints (CPU, memory and cardinality) of each server; (6.8,6.8′,6.9) allow us to count the number of migrations and state the limit on N (6.8 and 6.8′ together give a stronger linear relaxation than the single $a_{it} \geq x_{ijt} - x_{ijt-1}$); (6.10-6.11) keeps track of the change of states of the servers; (6.12) states the periods where a server has to be ON; (6.13–6.15) enforce the initial state (t = 0). The number of constraints of this model is dominated by the $n \times m \times h$ number of (6.8) and (6.3).

3.3 Lower Bound – An Extended Formulation

Solving large-sized instances of this application domain within short time limits is beyond the capability of exact algorithms. Therefore, one is generally forced to use an incomplete approach. Although an incomplete approach like large neighborhood search can usually find feasible solutions quickly, their qualities are often not evaluated as no bounds or provable approximation ratio can be found in the literature. Hence, it is important to be able to compute tighter lower bounds. In this section we present a column generation-based approach for computing a lower bound. Although we focus on a lower bound for the particular formulation (6), we believe it is generic enough to be relevant to other closely related problems of the literature that have at their core a series of cost-aware bin packing problems coupled with cost/migration constraints.

Let $b_{kt} \in \{0, 1\}$ be a variable for each bin packing of each time period to know whether the packing k is used for time period t. The set of all packings for period t is denoted by Ω_t. The packing k of period t is characterized by its cost c_{kt}, the server where each virtual machine is run and the state of each server. We use $x_{kijt} = 1$ if v_i is placed on s_j in the packing k at time period t and $o_{kjt} = 1$ if server s_j is ON in the packing k. In addition to b_{kt}, the variables $bto_{jt}, otb_{jt}, a_{it}$ and mig_{ijt} that we have already introduced for (6) are used in the column generation model (7). The restricted master problem is defined for a restricted number of packing variables ($\forall t \leq m,\ b_{kt} \in \Omega_t' \subset \Omega_t$):

$$\textbf{Minimize } z_4 = \sum_{t \in T}(\sum_{s_j \in SE}(Esta_j bto_{jt} + Esto_j otb_{jt}) + \sum_{k \in \Omega_t} c_{kt} b_{kt} + \sum_{v_i \in VM} Cmiga_{it})$$

$$\begin{array}{llll}
(7.1) & \sum_{k \in \Omega_t} b_{kt} & = 1 & (\forall\ p_t \in T) \quad (\lambda_t) \\
(7.2) & mig_{ijt} & \geq \sum_{k \in \Omega_t} x_{kijt} b_{kt} - \sum_{k \in \Omega_t} x_{k,i,j,t-1} b_{k,t-1} & (\forall\ v_i \in VM, s_j \in SE, p_t \in T) \quad (\pi_{ijt}) \\
(7.3) & a_{it} & \geq \sum_{s_j \in SE} mig_{ijt} & (\forall\ v_i \in VM, p_t \in T) \\
(7.4) & \sum_{v_i \in VM} a_{it} & \leq N & (\forall\ p_t \in T) \\
(7.5) & bto_{jt} & \geq \sum_{k \in \Omega_t} o_{kjt} b_{kt} - \sum_{k \in \Omega_t} o_{k,j,t-1} b_{k,t-1} & (\forall\ s_j \in SE, p_t \in T) \quad (\alpha_{jt}) \\
(7.6) & otb_{jt} & \geq \sum_{k \in \Omega_t} o_{k,j,t-1} b_{k,t-1} - \sum_{k \in \Omega_t} o_{kjt} b_{kt} & (\forall\ s_j \in SE, p_t \in T) \quad (\beta_{jt})
\end{array} \tag{7}$$

Let λ_t, π_{ijt}, α_{jt} and β_{jt} be the dual variables of constraints (7.1), (7.2), (7.5) and (7.6) respectively. We have h independent pricing problems and for each time period t we are looking for a negative reduced cost packing. The number of constraints (7.2) can prevent us from solving the relaxation of the master problem alone. We therefore turned to a relaxation of the migration constraints. The rationale is that the migration cost is really dominated by the server costs. Let $nmig_{jt} \in \mathbb{N}$ be the number of migrations occurring on server j and $u_{kjt} = \sum_{i \in VM} x_{kijt}$ the number of virtual machines allocated

to server j in the k-th packing of time t. We suggest removing the a and mig variables from formulation (7), adding the $nmig$ variables instead and replacing (7.2)–(7.4) by :

$$\begin{array}{lll} (7.2') \ nmig_{jt} \geq \sum_{k\in\Omega_t} u_{kjt}b_{kt} - \sum_k u_{k,j,t-1}b_{k,t-1} & (s_j \in SE, p_t \in T) & (\pi_{jt}) \\ (7.3') \ \sum_{j\in SE} nmig_{jt} \leq N & (\forall\ p_t \in T) & (\gamma_t) \end{array}$$

The last term in the objective is replaced by $Cmig(\sum_{t\in T}\sum_{j\in SE} nmig_{jt})$. The pricing problem for period t can now be seen as a cost-aware bin packing problem with an extra cost related to the number of items assigned to a bin and two side constraints: a cardinality and memory capacity constraint. The reduced cost r_{kt} of packing b_{kt} is equal to

$$r_{kt} = c_{kt} - \sum_{j\in SE}(o_{kjt}(-\alpha_{jt} + \alpha_{j,t+1} + \beta_{jt} - \beta_{j,t+1}) + u_{kjt}(-\pi_{jt} + \pi_{j,t+1})) - \lambda_t \tag{8}$$

For each bin j, the fixed and unit costs can be set to $f_j = Emin'_j - (-\alpha_{jt} + \alpha_{j,t+1} + \beta_{jt} - \beta_{j,t+1})$ and $c_j = \tau_j$ respectively. The cost depending on the number of items placed in bin j is denoted $\tau c_j = -(-\pi_{jt} + \pi_{j,t+1})$. Ignoring the constant term $-\lambda_t$ of the objective function, we summarize the pricing problem of period t by a CP model:

$$\begin{array}{ll} & \textbf{Minimize } r_t = \ ccpu + ccard \\ (9.1) & \textsc{BinPackingUsageCost}([x_1,\ldots,x_n],[cpu_1,\ldots,cpu_m],[y_1,\ldots,y_m],nbb,ccpu, \\ & \qquad [U_{1t},\ldots,U_{1n}],[(f_1,\tau_1),\ldots,(f_m,\tau_m)]) \\ (9.2) & \textsc{BinPackingUsageCost}([x_1,\ldots,x_n],[oc_1,\ldots,oc_m],\quad [y_1,\ldots,y_m],nbb,ccard, \\ & \qquad [1,\ldots,1],[(0,\tau c_1),\ldots,(0,\tau c_m)]) \\ (9.3) & \textsc{BinPacking}([x_1,\ldots,x_n],[mem_{1t},\ldots,mem_{mt}],nbb',[M_{1t},\ldots,M_{nt}]) \\ (9.4) & nbb \geq nbb' \\ (9.5) & \textsc{GlobalCardinality}([x_1,\ldots,x_n],[oc_1,\ldots,oc_m]) \\ (9.6) & y_j \begin{cases} = 1 & \text{if } p_t \in P_j \text{ or } f_j \leq 0 \\ \in \{0,1\} & \text{otherwise} \end{cases} \qquad (\forall\ s_j \in SE) \end{array} \tag{9}$$

Each variable $x_i \in SA_i$ gives the bin where item v_i is placed. $cpu_j \in [0, Umax'_j]$ and $mem_j \in [0, Mmax_j]$ encode the cpu and memory load of bin j, respectively. The number of items placed in bin j is given by $oc_j \in \{0,\ldots,n\}$ and $y_j \in \{0,1\}$ indicates if bin j *is ON* or not. The number of bins used is $nbb \in \{1,...,m\}$ (nbb' is an intermediate variable). Finally $ccpu \geq 0$ and $ccard \geq 0$ are real variables representing the costs related to CPU and cardinality. The costs are expressed by the state of the bins, thus matching the model of Section 2. A negative f_j is handled by pre-fixing y_j to 1 (constraint (9.6)).

Dual bound. The bottleneck of this method is the hardness of the pricing step, as proving that no negative reduced cost packing exist is unlikely to be tractable. At any iteration, if the optimal reduced costs $r^* = (r^*_1,\ldots,r^*_h)$ are known, a well-known lower bound of the linear relaxation of the master is $w_4 = z^*_4 + \sum_{t\in P_t} r^*_t$ where z^*_4 is the current optimal value of the **restricted** master at this iteration. Indeed, since r^*_t is the best reduced cost for period t, $\forall k \in \Omega_t,\ r_{kt} \geq r^*_t$, and using (8) we have the following:

$$\forall k \in \Omega_t,\ c_{tk} \geq r^*_t + \sum_{j\in SE}(o_{kjt}(-\alpha_{jt} + \alpha_{j,t+1} + \beta_{jt} - \beta_{j,t+1}) + u_{kjt}(-\pi_{jt} + \pi_{j,t+1})) + \lambda_t.$$

This shows that the solution $(\gamma,\pi,\alpha,\beta,\lambda + r^*)$ is dual feasible for the master which explains w_4. Now this reasoning also holds for any value smaller than r^*_t. Therefore

we still get a valid lower bound $\underline{w_4}$ if we use a lower bound $\underline{r_t^*}$ of each r_t^* and $\underline{w_4} = z_4^* + \sum_{t \in P_t} \underline{r_t^*} \leq w_4$. We note that this algorithm can therefore return a valid bound without succeeding in solving a single pricing problem to optimality. At the moment, the pricing problem is solved using a linear solver with a time-limit of three seconds so the best bound is used for $\underline{r_t^*}$ if the time limit is reached. This is critical for scaling with sub-problem size. We can always return the best $\underline{w_4}$ found over all iterations. In practice we terminate when the gap between $\underline{w_4}$ and z_4^* is less than 0.1%.

3.4 Upper Bounds

The EnergeTIC team initially designed a MIP model that was embedded in their platform but it failed to scale. The details of this model are not reported here. We investigated three different approaches for computing upper bounds. The first approach is the MIP model (6) of Section 3.2 which is an improvement of the model designed by EnergeTIC. The second approach which we call Temporal Greedy (TG) is currently employed in their platform. It proceeds by decomposing time and is more scalable. It greedily solves the problem period by period using model (6) restricted to one period (enforcing the known assignment of the previous period). Each time-period is used as a starting period as long as there is time left, and therefore, if required, the assignment is extended in both directions (toward the beginning and toward the end). The last one is a large neighborhood search (LNS)[14], which was originally developed for the machine reassignment problem of 2012 ROADEF Challenge which had only 1 time-period. Therefore we extended it in order to handle multiple time-periods.

4 Experimental Results

Cost-Aware Bin Packing Benchmarks. We first compare on randomly generated instances the lower bounds z_1^*, z_2^*, z_3^* as well as exact algorithms: Model (1), Arc-Flow Model (3) and a CP model using the BinPackingUsageCost constraint. Standard symmetry/dominance breaking techniques for BP are applied to the MIP [18] of Model (1) and CP [19]. A random instance is defined by (n, m, X), where n is the number of items ($n \in \{15, 25, 200, 250, 500\}$), m is the number of bins ($m \in \{10, 15, 25, 30\}$), and parameter $X \in \{1, 2, 3\}$ denotes that the item sizes are uniformly randomly generated in the intervals [1, 100], [20, 100], and [50, 100] respectively. The capacities of the bins are picked randomly from the sets {80, 100, 120, 150, 200, 250} and {800, 1000, 1200, 1500, 2000, 2500} when $n \in \{15, 25\}$ and $n \in \{200, 250, 500\}$ respectively. The fixed cost of each bin is set to its capacity and the unit cost is randomly picked from the interval [0, 1[. For each combination of $(n, m) \in \{(15, 10), (25, 15), (25, 25), (200, 10), (250, 15), (500, 30)\}$ and $X \in \{1, 2, 3\}$ we generated 10 instances giving 180 instances in total.

The time-limit was 600 seconds. If an approach failed to solve an instance within the time-limit then 600 was recorded as its solution time. All the experiments were carried out on a Dual Quad Core Xeon CPU, running Linux 2.6.25 x64, with 11.76 GB of RAM, and 2.66 GHz processor speed. The LP solver used was CPLEX 12.5 (default parameters) and the CP solver was Choco 2.1.5. Table 1 reports results for some classes due to lack of space. We report the average cpu time (denoted `cpu`) and the average

Table 1. Comparison of bounds obtained using MIP, Arc-Flow, CP, and Cutting-Stock approaches on random bin packing with usage cost problem instances with 600 seconds time-limit.

n	m	X	best ub	MIP				CP				Arc-Flow				Cutting-Stock
				z_1^*	ub	#nu	cpu	ub	#nu	cpu		z_3^*	ub	#nu	cpu	z_2^*
15	10	1	1005.2	956.8	1005.2	0 (0)	1.2	1005.2	0 (0)	**0.5**		959.6	1005.2	0 (0)	2.1	**960.3**
15	10	2	1267.4	1230.5	1267.4	0 (0)	1.1	1267.4	0 (0)	**0.2**		1244.5	1267.4	0 (0)	0.7	**1245.0**
15	10	3	1574.5	1522.3	1574.5	0 (0)	0.8	1574.5	0 (0)	0.7		1553.0	1574.5	0 (0)	**0.6**	**1553.5**
25	15	1	1665.6	1636.3	1665.6	0 (0)	35.1	1665.6	0 (0)	**24.0**		1638.9	1665.6	1 (0)	42.7	**1639.0**
25	15	2	2127.1	2086.4	2127.1	0 (0)	74.2	2127.1	0 (0)	**12.9**		2094.6	2127.1	0 (0)	61.2	**2094.9**
25	15	3	2682.8	2613.1	2682.8	0 (0)	22.6	2685.6	2 (0)	144.0		2657.9	2682.8	0 (0)	**11.3**	2657.9
500	30	1	32387.2	32187.0	32387.2	0 (0)	**18.1**	32387.2	0 (0)	57.6		32187.0	-	10 (10)	600	32187.0
500	30	2	40422.7	40235.8	40513.5	3 (0)	301.2	40422.7	**0** (0)	34.2		40235.8	-	10 (10)	600	40235.8
500	30	3	53395.6	53236.3	-	9 (2)	558.5	53395.6	**3** (0)	201.3		53236.3	-	10 (10)	600	53236.3

value of upper/lower-bounds found (denoted ub / z_x^*) (when a value is found for each instance of the class). Column #nu is a pair $x(y)$ giving the number of instances x (resp. y) for which an approach failed to prove optimality (resp. to find a feasible solution).

For the cutting-stock approach upper-bounds are not shown as the branch-and-price algorithm was not implemented. The CP approach shows better performance when scaling to larger size instances (and capacities) than the MIP and Arc-Flow models.

EnergeTIC Benchmarks. The industry partners provided 74 instances, where the maximum number of virtual machines (items), servers (bins), and time-periods are 242, 20 and 287 respectively.[4] The time-limit is 600 seconds. As mentioned previously, we compared three approaches for computing upper bounds: the MIP model, the Temporal Greedy approach (TG) currently used by EnergeTIC, and large neighborhood search (LNS) [14]. We also analyzed the lower bounds provided by the linear relaxation of the MIP model (LP), the best lower bound established by MIP when reaching the time-limit (MIP LB), and the bound provided by the linear relaxation of formulation (7) (CG). Table 2 gives an overview of the results by reporting (over the 74 instances) the average/median/max values of the **gap** to the best known bound[5], the **cpu** time, and the number of instances **#nu** when an approach fails to return any results within the time-limit. Table 2 also gives the results for a few hard instances.

Upper Bounds. Out of 74 instances, MIP was able to find solutions for 71 instances within the time-limit out of which 54 are proved optimal. It thus failed for 3 instances where the space requirement for CPLEX exceeded 11GB. Notice that the largest size instance has $1,389,080$ decision variables. Clearly, MIP-based systematic search cannot scale in terms of time and memory. TG is able to find solutions for 73 instances (so it failed on one instance), out of which 26 are optimal. Its quality deteriorates severely when one should anticipate expensive peaks in demand by placing adequately virtual machines several time periods before the peak. This can be seen in Table 2 where the maximum gap is 119.35%. LNS succeeds to find feasible solutions for all instances within 2 seconds, on average, but it was terminated after 600 seconds and for 41 in-

[4] The benchmarks are available from http://www.4c.ucc.ie/~dm6/energetic.tar.gz

[5] The gaps for lower and upper bounds are computed as $\frac{100\times(best\_ub-lb)}{best\_ub}$ and $\frac{100\times(ub-best\_lb)}{best\_lb}$) respectively. To compute mean/average/max values of gaps or time of a given approach, we exclude the instances where it fails to return any value (no feasible solution or a zero lower bound).

Table 2. Comparison between lower and upper bounds of the various approaches with 600 seconds time-limit (over 74 instances in the first part of the table and on a few specific instances in the second part).

			Lower bounds						**Upper bounds**				
			LP		**CG**		**MIP LB**		**LNS**	**MIP**		**TG**	
			gap	cpu	gap	cpu	gap	cpu	gap	gap	cpu	gap	cpu
		Mean	9,64	3,13	**0,32**	23,31	0,90	191,92	0,51	0,03	191,92	7,00	42,50
		Median	8,33	0,23	**0,10**	1,3	0	2,67	0	0	2,67	0,06	1,45
		Max	58,36	95,66	**7,14**	600	26,42	600	4,58	0,74	600	119,35	600
		#nu	3		**0**		3		**0**	3		1	
n	m	h	value	cpu	value	cpu	value	cpu	value	value	cpu	value	cpu
32	3	96	23492,8	9,5	**25404,7**	15,2	25043,6	600	25586,7	25575,7	600	36049,7	112,3
36	3	287	122831,3	4,5	**126716,8**	132,2	126597,9	600	127018,6	127654,4	600	127036,6	600
242	20	24	0	600	**37482,5**	600	0	600	40362,5	-	600	43027,6	14,2
242	20	24	0	600	**36890,8**	24,2	0	600	37701,6	-	600	36897,4	600
242	20	287	0	600	**431704,0**	600	0	600	439926,2	-	600	-	600
90	7	8	10420,7	14,4	**11431,9**	0,2	11236,3	600	11728,2	11435,3	600	11435,5	1,5

stances it found optimal solutions. Its average gap to the best known lower bound is less than 0.5% showing that LNS scales very well both in quality and problem size.

Lower bounds. The LP bound can be very far from the optimal value (its maximum gap is 58.36%) and does not scale since it fails on 3 instances even with 2 hour time-limit. The MIP obviously fails if the LP has failed. However, when solving the MIP, CPLEX automatically strengthens the formulation which allows us to solve many instances optimally where the LP bound was initially quite bad. Nevertheless, even after search there are cases where the gap can remain quite large (maximum of 26.42%). CG exhibits very good behaviour. Firstly, its gap clearly outperforms other bounds. Secondly, it can be stopped at any time and returns its current best master dual bound which is why #nu is 0 even though the time limit is reached on two cases (shown in Table 2). The first would improve to 38614.3 in 2000s whereas the second converges in 700 seconds without any improvement. Tables 2 shows that CG scales well both in terms of quality and size.

5 Conclusion and Future Work

Many optimisation problems in data centres can be described as a series of consecutive multi-dimensional Bin Packing with Usage Costs (BPUC) problems coupled by migration constraints and costs. First, we studied the lower bounds of a critical variant of bin packing for this domain that includes linear usage costs. We designed a CP approach that gives, so far, the best algorithm to solve BPUC exactly. Secondly, the usefulness of the exact algorithm and the efficient bounds for BPUC is shown within a column generation approach for the energy cost optimisation problem arising in data centres. These bounds are evaluated experimentally on real benchmarks and they assert the efficiency of the LNS approach [14] which was extended to handle consecutive BPUC problems.

The next step is to generalize the Martello and Toth bound L_2 [13] to the linear cost function which should improve the BinPackingUsageCost global constraint. We also plan to evaluate both column generation and LNS approaches on even larger size instances. We intend to solve the pricing problem with CP as we believe it can scale better for larger size problems.

References

1. A framework for data center energy productivity. Technical report, The green grid, 2008.
2. Efficience des Data Centers, les retombées du projet EnergeTIC. Technical report, http://www.vesta-system.cades-solutions.com/images/vestalis/4/energetic_white%20paper.pdf, 2013.
3. Shoshana Anily, Julien Bramel, and David Simchi-Levi. Worst-case analysis of heuristics for the bin packing problem with general cost structures. *Operational Research*, 42, 1994.
4. Mauro Maria Baldi, Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. The generalized bin packing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(6):1205 – 1220, 2012.
5. Teodor Gabriel Crainic, Guido Perboli, Walter Rei, and Roberto Tadei. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Comput. Oper. Res.*, 38(11):1474–1482, November 2011.
6. José M. Valério de Carvalho. Exact solution of bin packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86(0):629–659, 1999.
7. José M. Valério de Carvalho. LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253–273, 2002.
8. Corentin Dupont, Thomas Schulze, Giovanni Giuliani, Andrey Somov, and Fabien Hermenier. An energy aware framework for virtual machine placement in cloud federated data centres. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 4:1–4:10, New York, NY, USA, 2012. ACM.
9. Leah Epstein and Asaf Levin. Bin packing with general cost structures. *Math. Program.*, 132(1-2):355–391, April 2012.
10. Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 11:863–888, 1963.
11. Mohamed Haouari and Mehdi Serairi. Relaxations and exact solution of the variable sized bin packing problem. *Comput. Optim. Appl.*, 48(2):345–368, March 2011.
12. Chung lun Li and Zhi long Chen. Bin-packing problem with concave costs of bin utilization. *Naval Research Logistics*, 53:298–308, 2006.
13. Silvano Martello and Paolo Toth. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1):59 – 70, 1990.
14. Deepak Mehta, Barry O'Sullivan, and Helmut Simonis. Comparing solution methods for the machine reassignment problem. In Michela Milano, editor, *CP*, volume 7514 of *Lecture Notes in Computer Science*, pages 782–797. Springer, 2012.
15. Michele Monaci. *Algorithms for Packing and Scheduling Problems*. PhD thesis, Universit di Bologna, 2012.
16. Vinicius Petrucci, Orlando Loques, and Daniel Mosse. A dynamic configuration model for power-efficient virtualized server clusters. In *Proceedings of the 11th Brazilian Workshop on Real-Time and Embedded Systems*, 2009.
17. Thomas Rothvoss. Approximating bin packing within $O(\log OPT \cdot \log\log OPT)$ bins. Technical report, MIT, 2013.
18. Domenico Salvagnin. Orbital shrinking: A new tool for hybrid MIP/CP methods. In Carla Gomes and Meinolf Sellmann, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874 of *Lecture Notes in Computer Science*, pages 204–215. Springer Berlin Heidelberg, 2013.
19. Paul Shaw. A constraint for bin packing. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 648–662. Springer, 2004.
20. Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In *Proceedings of HotPower*, 2008.