Optimal energy-efficient placement of virtual machines with divisible sizes*

Gergely Halácsy and Zoltán Ádám Mann

Abstract

A key problem in the management of data centers is how to provision virtual machines based on the available physical machines, because an optimized placement can lead to significant reduction in energy consumption. This problem can be formulated as bin packing with heterogeneous bin types, where the cost of a bin depends on how full it is. We prove that under suitable conditions, an extended version of the First-Fit-Decreasing heuristic delivers optimal results for this problem.

1 Introduction

Data centers serve an ever-growing demand for computational power. The resulting growth in the energy consumption of data centers has both huge environmental impact and huge costs [19]. Therefore, the energy-efficient management of data centers has received much attention in the last couple of years.

Hardware vendors have devised techniques to reduce the power consumption of resources that are idle or lightly loaded [2]. As a result, the power consumption of a server depends significantly on its load. For example, the power consumption of a HP ProLiant DL380 G7 server varies from 280W (zero load) to 540W (full load) [10]. It is important to take into account the power characteristics (i.e., how power consumption depends on the server's load) when allocating workload on the servers. Most previous research assumed linear power characteristics [24], but some used more sophisticated power models, e.g., Hsu and Poole found the function $P(u) = P_{idle} + (P_{peak} - P_{idle}) \cdot u^{0.75}$ to yield the most accurate results [11].

A key technology for enabling effective data center management is virtualization. In a virtualized data center, the applications are deployed in virtual machines (VMs), which are in turn deployed on physical machines (PMs). This way, multiple applications can co-exist on the same PM in an isolated manner. Through virtualization, the workload can be allocated on a few highly utilized PMs and other PMs can be turned off, resulting in significant savings in energy consumption. This leads to an important optimization problem called the *VM allocation problem*: given the capacity and power consumption characteristics of the PMs and the load of the VMs, how to map the VMs on the PMs so that the overall energy consumption is minimal.

Several slightly different versions of the VM allocation problem have been addressed [16]. In many cases, the objective was to minimize the number of turned-on PMs as a proxy for energy consumption [3, 20, 22, 23]. This is only an approximation though since the power consumption of turned-on PMs can vary heavily. And with the advancement of technology, the dynamic power range (i.e., the difference between maximum and idle power consumption) of PMs is increasing.

In terms of the proposed algorithmic techniques, some works suggested exact methods but the majority applied heuristics. The proposed exact methods rely almost always on some form of mathematic programming (e.g., integer linear programming) and off-the-shelf solvers [9, 20]. Unfortunately, these approaches do not scale to practical problem sizes.

There is a natural connection between the VM allocation problem and bin-packing: VMs with their loads can be seen as items with given sizes that need to be packed into bins (PMs) of given capacity. On one hand, this proves that VM allocation is NP-hard. On the other hand, several researchers suggested to adopt bin-packing heuristics like First-Fit, Best-Fit, First-Fit-Decreasing etc. to the VM allocation problem [4, 14, 25]. However, VM placement is in several ways more complex than bin-packing, so that the known approximation results concerning these algorithms on bin-packing [6, 7] cannot be transferred to VM placement; in other words, they remain heuristics the results of which can be arbitrarily far from the optimum [15, 17, 18].

*This paper was published in: *Information Processing Letters, Volume 138, October 2018, Pages 51-56*

Some relevant generalizations of bin-packing have also been considered, mainly from an approximation point of view. Epstein and Levin showed an asymptotic polynomial-time approximation scheme (APTAS) for the problem in which different bin types are considered, each bin type is associated with a capacity and a cost, and the aim is to pack the items into a set of bins with minimum total cost, assuming that a sufficient number of bins are available from each type [8]. The problem that we are addressing now is more complex in the sense that the cost of a bin (its energy consumption) is not constant, but depends on how full it is. If each bin's cost depends linearly on how full it is, and this linear function is the same for each bin, then it is easy to see that the problem admits an APTAS [17]. However, the problem in which the cost of different bins can be described by different and not necessarily linear functions (which is a more realistic model of the VM allocation problem), has to our knowledge not been addressed yet.

The hardness of bin-packing arises partly because arbitrary sizes can appear in the input. Constraints on the allowed sizes can make the problem much easier. If the item sizes form a *divisible* series – i.e., for each pair of different item sizes, the smaller is a divisor of the larger – bin-packing can be solved optimally in polynomial time [5]. In this paper, we use a similar approach to devise a polynomial-time exact algorithm for the generalized problem in which the cost of bins depends on how full they are, under the assumption that the item sizes form a divisible series. This assumption is not too strict for VM allocation, because PM capacities and VM sizes are often powers of 2 [21], leading to divisible item size series.

The contributions of this paper are as follows:

- We address a version of the VM allocation problem, aiming to minimize total power consumption, where the power consumption of each type of PM is given by some function of its load. In contrast to previous works, we make only very light assumptions on these functions: they must be monotonously increasing and concave. These assumptions hold for example for the linear power consumption characteristics used by several researchers [1, 24] and also for the more sophisticated power consumption characteristics proposed by Hsu and Poole [11]. Further, we assume that VM sizes form a divisible series. This assumption holds for example if the VM sizes are all powers of two, which occurs frequently in practice, as reported by Shen et al. [21].
- We devise an algorithm called OptDiv for this problem. OptDiv is based on the First-Fit-Decreasing (FFD) heuristic, but FFD is oblivious of bin costs, so OptDiv extends it with informed decisions relating to power consumption.
- We prove that, under the given assumptions, OptDiv results in optimal total power consumption.
- OptDiv is very fast, in contrast to the proposed APTASs that are often not practical because of the huge constants in their execution times [13].

2 Preliminaries

In the *bin-packing* problem, we are given a set of n items with sizes $s_1 \leq s_2 \leq \ldots \leq s_n$ and a sufficiently large set of bins, each having capacity C, where $s_n \leq C$. The aim is to pack all the items into a minimal number m of bins $B_1, \ldots, B_m$ so that for each B_i, the sum of the sizes of the items in B_i is at most C. B_i denotes both the ith bin and the set of items in that bin.

The items arrive in some order $s_{i_1}, \ldots, s_{i_n}$. The *First-Fit* heuristic (FF) processes the items in this order: it starts by opening a bin B_1 for item s_{i_1}. In a general step, when bins $B_1, \ldots, B_k$ are already in use and the next item s_{i_j} is considered, the algorithm puts s_{i_j} into the first bin from $B_1, \ldots, B_k$ where it fits if there is such a bin, otherwise it opens a new bin B_{k+1}. The *First-Fit-Decreasing* heuristic (FFD) consists of first sorting the items in non-increasing order of their size, and then performing FF in this order of the items. Both FF and FFD are known to be approximation algorithms for the bin-packing problem [6, 7].

Let x, y be arbitrary real numbers. We call x a divisor of y and y a multiple of x if there exists an integer z such that $x \cdot z = y$. This relation is denoted as $x \mid y$. The input of bin-packing is *weakly divisible* if $s_i \mid s_j$ holds for all $1 \leq i < j \leq n$. If, in addition, also $s_n \mid C$ holds, then the input is *strongly divisible*.

The following known results about divisible inputs to bin-packing will be important.

Lemma 1 ([5], Fact 2). *Let s_i be an item size in a weakly divisible input and let T be a subset of the items such that no item in T has size larger than s_i but the sum of the sizes of the items in T is at least s_i. Then, there is a subset $T' \subseteq T$ such that the sum of the sizes of the items in T' equals s_i.*

Theorem 2 ([5], Theorem 2). *For weakly divisible inputs, the FFD algorithm always uses the minimal number of bins.*

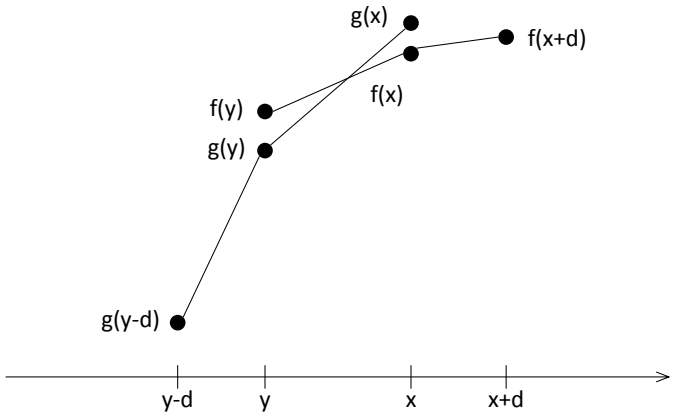

Figure 1: To the proof of Lemma 4

In the *VM allocation* problem, the input contains, just like in bin-packing, a set of n items with sizes $s_1 \leq s_2 \leq \ldots \leq s_n$. In addition, there are k bin types $T_1, \ldots, T_k$. Each bin has capacity C, where $s_n \leq C$. Each bin type T_i is characterized by a cost function $F_i : [0, C] \to \mathbb{R}_+$. For a bin B, let $\Sigma(B)$ denote the sum of the sizes of the items packed into B; this will be called the *size of the bin*. If B is a bin of type T_i, then the cost of B is $\gamma(B) = F_i(\Sigma(B))$. From each bin type, a sufficient number of bins is available. The objective is to pack all the items into bins $B_1, \ldots, B_m$ so that for each B_j, $\Sigma(B_j) \leq C$, and the total cost of the packing $\sum_{j=1}^{m} \gamma(B_j)$ is minimal.

For a size $0 \leq x \leq C$, a cost function that assigns minimal cost to size x among the functions $F_1, \ldots, F_k$ will be called an *optimal cost function* for the given size x.

3 Solving the VM allocation problem for divisible inputs

3.1 Proposed algorithm (OptDiv)

Given an instance of the VM allocation problem with item sizes $s_1, \ldots, s_n$, capacity C, and bin type cost functions $F_1, \ldots, F_k$, our algorithm OptDiv consists of the following two phases:

1. Running FFD with item sizes $s_1, \ldots, s_n$ and capacity C. The result is a series of disjoint subsets $B_1, \ldots, B_m$ such that in each B_i, $\Sigma(B_i) \leq C$.

2. Selecting for each B_i a bin of the type j for which $F_j(\Sigma(B_i))$ is minimal, i.e., F_j is optimal for size $\Sigma(B_i)$.

In other words, we disregard the cost functions in the first phase and pack the items in a fictive set of unit-cost bins. In the second phase, the real bins are chosen based on their costs, choosing for each subset of items forming a fictive bin a real bin that has (locally) optimal costs for the given subset of items.

3.2 Optimality

It is clear that OptDiv returns a valid solution to the VM allocation problem, but it is not clear that this solution is an optimal one. Our main result is the following theorem:

Theorem 3. *If the input is weakly divisible and the F_j functions are monotonously increasing and concave, then OptDiv yields optimal results for the VM allocation problem.*

Before proving Theorem 3, some lemmas are needed.

Lemma 4. *Let B and B' be two bins with $\Sigma(B) = x$ and $\Sigma(B') = y$, where $x > y$. Let f be an optimal cost function for bin size x and g an optimal cost function for bin size y. Let $d > 0$ be such that $x + d \leq C$ and $y - d \geq 0$. Assume that f and g are monotonously increasing and concave. Then,*

$$f(x + d) + g(y - d) \leq f(x) + g(y).$$

Proof. Since f is optimal for x and g is optimal for y, we have $f(x) \leq g(x)$ and $g(y) \leq f(y)$. Therefore,

$$\frac{f(x) - f(y)}{x - y} \leq \frac{g(x) - g(y)}{x - y}. \tag{1}$$

Since f is concave, the slope of the chord in $[y, x]$ is at least as high as the slope of the chord in $[x, x+d]$ (see Figure 1):

$$\frac{f(x) - f(y)}{x - y} \geq \frac{f(x+d) - f(x)}{d}. \tag{2}$$

Since g is concave, the slope of the chord in $[y, x]$ is at most as high as the slope of the chord in $[y-d, y]$:

$$\frac{g(x) - g(y)}{x - y} \leq \frac{g(y) - g(y-d)}{d}. \tag{3}$$

Combining inequalities (1)-(3), we get:

$$\frac{f(x+d) - f(x)}{d} \leq \frac{f(x) - f(y)}{x - y} \leq \frac{g(x) - g(y)}{x - y} \leq \frac{g(y) - g(y-d)}{d},$$

and hence

$$\frac{f(x+d) - f(x)}{d} \leq \frac{g(y) - g(y-d)}{d}.$$

Multiplying by d and reordering gives the stated inequality. □

The following shows that Lemma 4 also holds if $x = y$ and $f = g$.

Lemma 5. *Let B and B' be two bins with $\Sigma(B) = \Sigma(B') = x$. Let f be an optimal cost function for bin size x. Let $d > 0$ be such that $x + d \leq C$ and $x - d \geq 0$. Assume that f is monotonously increasing and concave. Then,*

$$f(x+d) + f(x-d) \leq 2 \cdot f(x).$$

Proof. Since f is concave, the slope of the chord in $[x-d, x]$ is at least as high as the slope of the chord in $[x, x+d]$:

$$\frac{f(x) - f(x-d)}{d} \geq \frac{f(x+d) - f(x)}{d}. \tag{4}$$

Multiplying by d and reordering gives the stated inequality. □

Lemma 6. *Let $B_1, \ldots, B_m$ be the list of subsets as returned by FFD at the end of Step 1 of the algorithm, in the order as FFD opened them. If the input is weakly divisible, then $\Sigma(B_1) \geq \Sigma(B_2) \geq \ldots \geq \Sigma(B_m)$.*

Proof. We prove by induction that the required property holds throughout the operation of the FFD algorithm. Assume that so far it holds and now the item with size s_i is processed, and the algorithm decides to put it in bin B_j, leading to $B'_j = B_j \cup \{s_i\}$. If $j = 1$, then the desired property cannot be violated. Otherwise, we need to show that $\Sigma(B_{j-1}) \geq \Sigma(B'_j)$.

Since the input is weakly divisible and the items are processed in non-increasing order, s_i is a divisor of the size of all items that were packed before. As a consequence, $s_i \mid \Sigma(B_{j-1})$ and $s_i \mid \Sigma(B_j)$. Since FFD chooses the first fitting bin and it chose B_j and not B_{j-1}, it follows that $\Sigma(B_{j-1}) > \Sigma(B_j)$. Together with the fact that both are multiples of s_i, this implies that $\Sigma(B_{j-1}) \geq \Sigma(B_j) + s_i$, and hence $\Sigma(B_{j-1}) \geq \Sigma(B'_j)$. □

Now everything is in place to prove our main theorem.

Proof of Theorem 3. We start by describing a method that assigns a *code* to any solution of the VM allocation problem. The method first sorts the bins in non-increasing order of their size, and encodes them in this order. For encoding the contents of a bin, the contained items are also sorted in non-increasing order of their size, and their sizes are enumerated in this order. The encoding of consecutive bins is separated by a special delimiter symbol ($*$). E.g., the code $(6, 6, 3, *, 6, 3)$ encodes a packing of five items into two bins, where the first bin contains two items of size 6 and one item of size 3, whereas the second bin contains one item of size 6 and one item of size 3. Any solution has a definite code, but multiple solutions can have the same code, since the code does not contain information about the bin types and it does not differentiate between items of the same size.

In order to prove the theorem, we use proof by contradiction: we assume that the solution found by OptDiv is not optimal. Let the solution delivered by OptDiv, denoted by A, consist of the bins $B_1, \ldots, B_m$ (in the order as returned by FFD in the first step). Consider an optimal solution O with bins $O_1, \ldots, O_t$ (in non-increasing order of their size), such that the code of O has the longest prefix in common with the code of A.

The solutions A and O have the following properties:

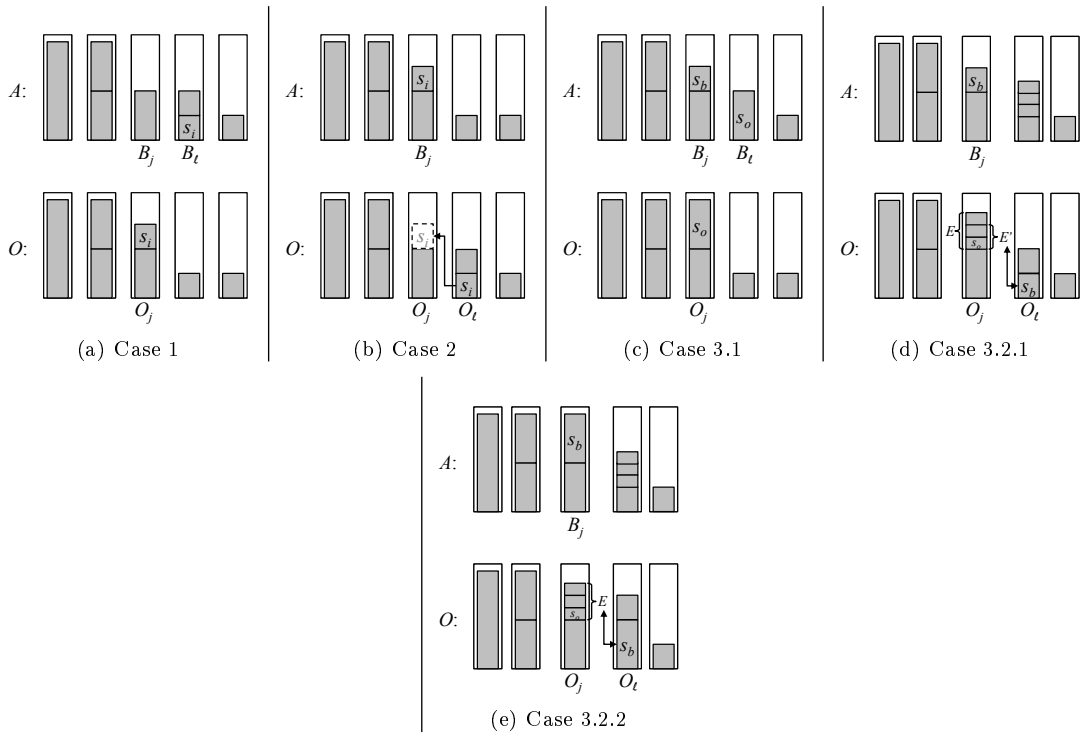

Figure 2: To the proof of Theorem 3

1. Because of Lemma 6, we can assume that the code of A contains the encoding of bins $B_1, \ldots, B_m$ in this order.
2. Because of Theorem 2, $m \leq t$.
3. Since O is optimal, it uses for each bin a bin type, the cost function of which is optimal for the given bin size. The same holds also for A because of the way it is created (second step of the algorithm).
4. According to the indirect assumption, the total cost of O is strictly less than the total cost of A.
5. Since the cost functions assign positive values to even an empty bin and O is optimal, O does not contain empty bins. The same holds also for A because of the way it is created.
6. If there are multiple items with the same size, the code does not differentiate between them. We can assume that they appear in the same order in A and O. E.g., if there are two items with size 6, then the first 6 in the code of A and the first 6 in the code of O represent the same item.
7. It can be assumed without loss of generality that, for any two bins with equal size, the same bin type is used.

Let j be the index of the first bin where the codes of A and O differ. Clearly, $1 \leq j \leq m$: if j were greater than m (i.e., the code of A were a prefix of the code of O), that would mean that the first m bins of O contain all items and O has further bins, which would contradict property 5. Because of property 2, also $j \leq t$ follows.

Based on the difference between B_j and O_j, we can differentiate three cases. Our aim is to show a contradiction in each of the three cases.

Case 1 (see Figure 2(a)): B_j is a proper subset of O_j. This means that O_j contains all items that B_j contains plus at least one more item s_i. Since this is the first difference between A and O, the item s_i must be in the solution A in a bin B_ℓ, where $\ell > j$. However, this contradicts the FFD logic: when the FFD algorithm packed item s_i, it should have put it into bin B_j, since it would fit into that bin as well (as demonstrated by the fact that $B_j \cup \{s_i\} \subseteq O_j$) and FFD always chooses the first fitting bin.

Case 2 (see Figure 2(b)): O_j is a proper subset of B_j. Let s_i be the first element in the code of B_j that is not contained in O_j. In solution O, the item s_i must be in a bin O_ℓ, where $\ell > j$. Now let us move item s_i from O_ℓ to O_j, resulting in a new solution O'. This is indeed a solution, since s_i fits into bin j, as demonstrated by the fact that $O_j \cup \{s_i\} \subseteq B_j$. Recall that the bins of O are in non-increasing order of their size, and so $\Sigma(O_j) \geq \Sigma(O_\ell)$. If

$\Sigma(O_j) > \Sigma(O_\ell)$, then Lemma 4 (cf. also property 3), if $\Sigma(O_j) = \Sigma(O_\ell)$, then Lemma 5 (cf. also property 7) shows that the total cost of solution O' is less than or equal to the total cost of solution O. Since O is optimal, it follows that O' is also optimal. On the other hand, the new position of item s_i means that the prefix of the code of O' that is in common with the code of A is at least one longer than the prefix of the code of O that is in common with the code of A, contradicting the choice of O.

Case 3: neither $B_j \subsetneq O_j$ nor $O_j \subsetneq B_j$. Since $B_j \neq O_j$ and because of property 6, this means that the codes of B_j and O_j are equal up to some point, but continue with different elements; let these be s_b in B_j and s_o in O_j, where $s_b \neq s_o$. We have two sub-cases according to the relation between s_b and s_o.

Case 3.1 (see Figure 2(c)): $s_b < s_o$. Since this is the first difference between A and O and because the item s_o cannot appear later in the code of B_j, the item s_o must be in the solution A in a bin B_ℓ, where $\ell > j$. However, this contradicts the FFD logic: when the FFD algorithm packed item s_o, s_b was not yet in B_j, so that s_o would have fit into B_j (just as it fits into O_j), and FFD always chooses the first fitting bin.

Case 3.2: $s_b > s_o$. Since this is the first difference between A and O and because the item s_b cannot appear later in the code of O_j, the item s_b must be in the solution O in a bin O_ℓ, where $\ell > j$. Let D denote the set of items corresponding to the maximal common prefix in the code of O_j and B_j, and let E denote the further elements in O_j. We must differentiate between two sub-cases according to the size of E.

Case 3.2.1 (see Figure 2(d)): $\Sigma(E) \geq s_b$. Since s_o is the greatest element in E, all elements in E are less than s_b. Therefore, Lemma 1 can be applied, and it guarantees the existence of a subset $E' \subseteq E$ with $\Sigma(E') = s_b$. Now, we can swap item s_b in O_ℓ with the items forming E' in O_j, resulting in a new solution O'. Since $\Sigma(E') = s_b$, this swap does not alter the size of the bins. As a consequence, O' is also an optimal solution. On the other hand, now s_b has become a common element in the jth bin of A and O', so that the prefix of the code of O' that is in common with the code of A is at least one longer than the prefix of the code of O that is in common with the code of A, contradicting the choice of O.

Case 3.2.2 (see Figure 2(e)): $\Sigma(E) < s_b$. Let $d = s_b - \Sigma(E) > 0$. We swap item s_b in O_ℓ with the items forming E in O_j, resulting in a new solution O'. This is indeed a solution: the size of O_ℓ decreased by d, while the size of O_j increased by d to $\Sigma(D) + s_b$, which is still below the bin capacity since $D \cup \{s_b\} \subseteq B_j$. Recall that the bins of O are in non-increasing order of their size, and so $\Sigma(O_j) \geq \Sigma(O_\ell)$. If $\Sigma(O_j) > \Sigma(O_\ell)$, then Lemma 4 (cf. also property 3), if $\Sigma(O_j) = \Sigma(O_\ell)$, then Lemma 5 (cf. also property 7) shows that the total cost of solution O' is less than or equal to the total cost of solution O. Since O is optimal, it follows that O' is also optimal. On the other hand, the new position of item s_b means that the prefix of the code of O' that is in common with the code of A is at least one longer than the prefix of the code of O that is in common with the code of A, contradicting the choice of O.

Each case led to a contradiction, thus completing the proof. □

3.3 Execution time

Now we investigate the asymptotic running time of the proposed algorithm. Recall that n denotes the number of items to pack and k denotes the number of bin types available.

The first step is an invocation of FFD on the set of items. FFD can be implemented to run in $\Theta(n \cdot \log n)$ time [12].

In the second step, the optimal bin type has to be found for each subset returned by FFD. This requires evaluating each of the k cost functions for each of the m subsets. Assuming that the function evaluations can be done in $O(1)$ time, and using that $m \leq n$, the time required for the second step is $O(n \cdot k)$.

Thus, the overall time complexity of the algorithm is $O(n \cdot (k + \log n))$.

4 Empirical evaluation

To empirically assess the effectiveness and efficiency of the proposed algorithm (OptDiv), we implemented it in a C++ program together with two alternatives. One of them is the normal – i.e., power-oblivious – FFD algorithm, the other is the Power-Aware Best Fit Decreasing (PABFD) algorithm proposed in the literature specifically for the VM allocation problem [4].

We fixed the capacity of the PMs to 1024 and generated n VMs with sizes chosen uniformly at random from the powers of two between 1 and 1024, thus ensuring the divisibility of the VM sizes. We considered three PM types, each with different linear power characteristics: (1) from 150 to 600W, (2) from 300 to 350W, and (3) from 250 to 450W. The measurements were carried out on a notebook computer with Intel i3-3110M CPU running at 2.40 GHz with 4GB RAM. Each reported result is the average of 10 runs.

Table 1: Results of the empirical evaluation. m: number of PMs; kW: total power consumption in kW; $W\%$: total power consumption relative to OptDiv; t: execution time in msec; $t\%$: execution time relative to OptDiv

	100 VMs					500 VMs					1,000 VMs					5,000 VMs					10,000 VMs				
Algorithm	m	kW	$W\%$	t	$t\%$	m	kW	$W\%$	t	$t\%$	m	kW	$W\%$	t	$t\%$	m	kW	$W\%$	t	$t\%$	m	kW	$W\%$	t	$t\%$
OptDiv	20.5	7.1	100	0.5	100	90	31.5	100	3	100	183	63.8	100	9.3	100	924	323	100	180	100	1820	637	100	596	100
FFD	20.5	9.5	134	0.4	80	90	42.2	134	2.4	80	183	84.6	133	8.3	89	924	430	133	171	95	1820	849	133	581	98
PABFD	20.5	8.0	113	0.2	40	90	37.0	118	3.7	123	183	75.4	118	13.4	144	924	380	118	286	159	1820	751	118	1021	171

Table 1 shows the results for varying number of VMs, according to multiple metrics. Concerning the number of PMs, the three algorithms perform equal: each uses the minimum number of PMs in accordance with Theorem 2. Regarding power consumption, however, there are clear differences: FFD leads to 33-34%, PABFD to 13-18% higher power consumption than OptDiv. In terms of execution time, FFD is the fastest, but the relative performance penalty for OptDiv diminishes for larger instances. The execution time of PABFD is growing faster than that of the other two algorithms, leading to 71% higher execution time than OptDiv for 10,000 VMs, but even that is just about one second.

Overall, the experiments reinforced the usefulness of the proposed OptDiv algorithm: it is very fast and delivers significant reductions in power consumption compared to competing algorithms.

5 Open problems

Several interesting open problems remain. In particular, it is unclear if the presented methods can be carried over to the following generalizations of the problem: (i) bin types with different capacities; (ii) limited number of available bins per bin type; (iii) multi-dimensional vector bin packing.

Acknowledgements

This work was partially supported by the Hungarian Scientific Research Fund (Grant Nr. OTKA 108947) and by the European Union's Horizon 2020 research and innovation programme under grant 731678 (RestAssured).

References

[1] Ehsan Ahvar, Shohreh Ahvar, Zoltán Ádám Mann, Noel Crespi, Joaquin Garcia-Alfaro, and Roch Glitho. CACEV: a cost and carbon emission-efficient virtual machine placement method for green distributed clouds. In *Proceedings of the 13th IEEE International Conference on Services Computing*, pages 275–282, 2016.

[2] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[3] Dávid Bartók and Zoltán Ádám Mann. A branch-and-bound approach to virtual machine placement. In *Proceedings of the 3rd HPI Cloud Symposium "Operating the Cloud"*, pages 49–63, 2015.

[4] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[5] E.G Coffman, M.R Garey, and D.S Johnson. Bin packing with divisible item sizes. *Journal of Complexity*, 3(4):406–428, 1987.

[6] György Dósa. The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leq 11/9OPT(I) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, pages 1–11. Springer, 2007.

[7] György Dósa and Jiří Sgall. First fit bin packing: A tight analysis. In *30th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 538–549, 2013.

[8] Leah Epstein and Asaf Levin. An APTAS for generalized cost variable-sized bin packing. *SIAM Journal on Computing*, 38(1):411–428, 2008.

[9] Brian Guenter, Navendu Jain, and Charles Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *Proceedings of IEEE INFOCOM*, pages 1332–1340. IEEE, 2011.

[10] HP. Power efficiency and power management in HP ProLiant servers. `http://h10032.www1.hp.com/ctg/Manual/c03161908.pdf`, 2012.

[11] Chung-Hsing Hsu and Stephen W. Poole. Power signature analysis of the SPECpower_ssj2008 benchmark. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 227–236, 2011.

[12] David S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272–314, 1974.

[13] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science*, pages 312–320. IEEE, 1982.

[14] Wubin Li, Johan Tordsson, and Erik Elmroth. Virtual machine placement for predictable and time-constrained peak loads. In *Proceedings of the 8th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON 2011)*, pages 120–134. Springer, 2011.

[15] Zoltán Ádám Mann. *Optimization in computer engineering – Theory and applications*. Scientific Research Publishing, 2011.

[16] Zoltán Ádám Mann. Allocation of virtual machines in cloud data centers – a survey of problem models and optimization algorithms. *ACM Computing Surveys*, 48(1), 2015.

[17] Zoltán Ádám Mann. Approximability of virtual machine allocation: much harder than bin packing. In *Proceedings of the 9th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*, pages 21–30, 2015.

[18] Zoltán Ádám Mann. Rigorous results on the effectiveness of some heuristics for the consolidation of virtual machines in a cloud data center. *Future Generation Computer Systems*, 51:1–6, 2015.

[19] Natural Resources Defense Council. Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers. `http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf`, 2014.

[20] B. C. Ribas, R. M. Suguimoto, R. A. N. R. Montano, F. Silva, L. de Bona, and M. A. Castilho. On modelling virtual machine consolidation to pseudo-Boolean constraints. In *13th Ibero-American Conference on AI*, pages 361–370, 2012.

[21] Siqi Shen, Vincent van Beek, and Alexandru Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 465–474, 2015.

[22] L. Shi, J. Furlong, and R. Wang. Empirical evaluation of vector bin packing algorithms for energy efficient data centers. In *IEEE Symposium on Computers and Communications*, pages 9–15, 2013.

[23] W. Song, Z. Xiao, Q. Chen, and H. Luo. Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 63(11):2647–2660, 2014.

[24] P. Svärd, W. Li, E. Wadbro, J. Tordsson, and E. Elmroth. Continuous datacenter consolidation. In *IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 387–396, 2015.

[25] Luis Tomás and Johan Tordsson. An autonomic approach to risk-aware data center overbooking. *IEEE Transactions on Cloud Computing*, 2(3):292–305, 2014.