Noname manuscript No.
(will be inserted by the editor)

Scalable and Automatic Virtual Machines Placement based on Behavioral Similarities

Claudia Canali · **Riccardo Lancellotti**

the date of receipt and acceptance should be inserted later

Abstract The success of the cloud computing paradigm is leading to a significant growth in size and complexity of cloud data centers. This growth exacerbates the scalability issues of the Virtual Machines (VMs) placement problem, that assigns VMs to the physical nodes of the infrastructure. This task can be modelled as a multi-dimensional bin-packing problem, with the goal to minimize the number of physical servers (for economic and environmental reasons), while ensuring that each VM can access the resources required in the next future. Unfortunately, the naïve bin packing problem applied to a real data center is not solvable in a reasonable time because the high number of VMs and of physical nodes makes the problem computationally unmanageable. Existing solutions improve scalability at the expense of solution quality, resulting in higher costs and heavier environmental footprint. The *Class-Based placement* technique (CBP) is a novel approach that exploits existing solutions to automatically group VMs showing similar behaviour. The Class-Based technique solves a placement problem that considers only some representative VMs for each class, and that can be replicated as a *building block* to solve the global VMs placement problem. Using real traces, we analyse our proposal performance, comparing different alternatives to automatically determine the number of building blocks. Furthermore, we compare our proposal against the existing alternatives and evaluate the results for different workload compositions. We demonstrate that the CBP proposal outperforms existing solutions in terms of scalability and VM placement quality.

1 Introduction

The success of cloud computing is clearly testified by the expected growth in fifteen years by two order of magnitude in terms of data stored and processed in cloud systems [1]. The reason behind this success is the on-demand, pay-as-you-go philosophy

C. Canali, R. Lancellotti (✉)
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
E-mail: {claudia.canali, riccardo.lancellotti}@unimore.it

used by cloud computing providers to provide computational, storage and networking resources for their customers. The offered flexibility is ideal to face the highly variable workloads of novel services while reducing the costs associated to the management of the ICT infrastructure. This success is determining a trend towards the deployment of larger and even more powerful cloud data centers, hosting an always increasing number of Virtual Machines (VMs). However, as data centers grow in size and complexity, new challenges arise for automated processes of system monitoring and management. In this paper we explicitly focus on the management problem of VMs placement, that is responsible for allocating each VM to a physical node of the infrastructure with the aim to minimize the data center energy consumption to reduce costs and environmental footprint [2,3]. The typical placement problem formulation is that of a multi-dimensional bin packing: the optimization goal is to minimize the number of physical nodes required to host the VMs, while the problem constraints captures the time-varying requirements of multiple resources (e.g., CPU, memory, network traffic) for each VM at different future time intervals [4,5]. The nature of the multi-dimensional bin-packing problem is *NP-hard*: hence, as the number of VMs grows, it is not possible to reach an optimal solution in a reasonable time. Existing solutions typically address the scalability issues of the bin-packing problem introducing strong simplifications and adopting heuristics to reduce problem dimensionality and computational costs.

A widely adopted solution to reduce the dimensionality of the VMs placement problem is to consider the nominal requirements of the VMs [6–8]. If the VMs size is a sub-multiple of the physical node capacity, the VMs placement becomes a linear problem that is easily solvable in a short time. Furthermore, we do not need to consider the evolution over time of the actual VMs resource requirements, adding a further simplification to the placement problem. The main drawback of this approach is the overestimation of the actual resource requirements of the VMs, whose utilization is typically below 100% [9]. This leads to an inefficient use of the cloud data center, with a higher than needed number of physical nodes used for the overall infrastructure. A second solution to reduce the VMs placement dimensionality is to limit the number of resources that are considered in the bin packing problem and/or the number of time intervals considered as constraints of the optimization problem [10,5]: for example, in many cases just the peak CPU usage over a whole day (24 hours) is taken into account. However, this solution fails to take advantage from workload complementary patterns (e.g., co-presence of applications with diurnal and nocturnal peak utilization), and tends to result in sub-optimal solutions. Furthermore, even with these simplifications the computational cost for solving the VM placement problem remains high, especially for large data centers, to the extent that the time to obtain a feasible solution may be not acceptable for the dynamic management of the infrastructure. Heuristics can be exploited to reduce the computational demand of the VM placement problem [11,12], but these solutions are feasible only when jointly used with dimensionality reduction, thus hindering the use of multiple time intervals with a negative effect on the solution quality. We can summarize that state of the art approaches for the VMs placement problem typically fail to consider the actual behavior of VMs and/or rely on simple heuristics, producing in both cases low quality solutions that lead to a waste of cloud data center resources.

In this paper we propose a novel approach for VMs placement, namely *Class-Based Placement* (CBP), that exploits the presence of *classes* of VMs with a similar behavior in terms of resource usage within the cloud data center. Information about behavior similarity can be either inferred from a PaaS vision of the cloud infrastructure that knows the applications running on the VMs or can be obtained applying recent methodologies to cluster together similar VMs [13–15] in cloud systems. Our technique shifts the point of view from a single bin-packing problem, that considers the whole data center, to a much smaller problem, limited to a few representative VMs for each class, that can be replicated as a building block to create the solution for the global VM placement problem. The small size of the building-block problem can be solved in short time even taking into account an amount of data and constraints that would not be possible to consider in the global bin-packing problem. Our analysis specifically addresses the issue of automatically determining which is the most suitable size of the building block problem, proposing and comparing two different alternatives to split the global problem. Our claim is that the CBP proposal can effectively reduce the scalability issues of the VMs placement by reducing the size of the problem. Moreover, higher quality placement solutions can be achieved compared to existing alternatives since the better scalability allows to take into account a more complex model of the VM resource demand. A first version of this technique was proposed in [16]. However, this paper represents a significant step ahead with respect to that previous study for a twofold reason: first, it addresses the open issue of automatically identifying the best number of building blocks to use to split the global problem; second, it presents a wider range of experiments, including a sensitivity analysis of the technique performance with respect to different workload compositions.

A thorough experimental evaluation based on traces from a real data center analyzes the performance of our proposal in terms of scalability and quality of placement solution: in other words, we evaluate the capability of the CBP placement to solve large problems and to minimize the number of used physical nodes. We compare our proposal with state of the art models for VMs placement [4]. Our experiments achieve the following main results: (1) we evaluate the pros and cons of different approaches to select the number and size of building blocks, and we identify the best option; (2) we show the scalability problems of existing solutions for VMs placement in cloud data centers; (3) we demonstrate that our proposal outperforms existing techniques from both the points of view of resolution time and quality of the solution; (4) we show that our proposal can effectively leverage the presence of complementary workload patterns.

The remainder of this paper is organized as follows. Section 2 describes the reference scenario for our proposal, while Section 3 describes our model for solving the VM placement problem. Section 4 describes the results of the methodology evaluation. Finally, Section 5 discusses the related work and Section 6 concludes the paper with some final remarks.

2 Management of IaaS Cloud Data Centers

We now describe the reference scenario, that is used to outline the characteristics of the proposed technique for the management of a cloud data center. In particular, we focus on the operations that decide the placement of the VMs over the physical nodes of the infrastructure, depicted in Figure 1.

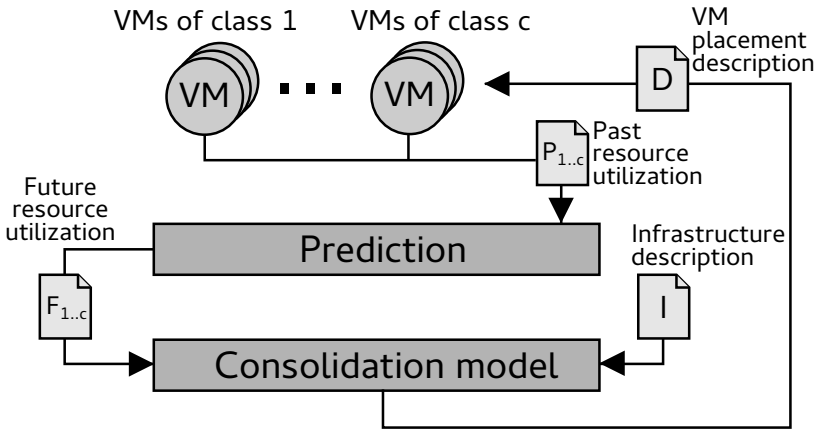


Fig. 1: VMs placement in a IaaS data center

In our proposal we make the following two assumptions:

- VMs placement is a periodic task that aims at mapping VMs over the physical nodes of the infrastructure with the goal of minimizing the number of used nodes, while satisfying the requirements of each VM in terms of resource usage:
- we can group VMs into classes with similar behavior; VMs belonging to the same class exhibit similar resource requirements.

The second condition typically occurs when a customer application is replicated over a distributed architecture for scalability and availability reasons: in this case, a dispatcher distributes the requests among the VMs running the same software component of a same customer application, with the goal of balancing the application load. The dispatcher action ensures that the VMs of same class exhibit a similar behavior in terms of resource requirements [17]. If the balancing dispatcher is provided by the cloud platform (e.g., the Elastic load balancing mechanism[1]) the cloud provider has already an information of which VMs belongs to the same customer application. In the case of a pure IaaS provider, where the load balancing is configured by the cloud customer, we can rely on methodologies to automatically cluster VMs with similar behavior in terms of resource usage without having any knowledge of the software component they run. To this purpose, methodologies based on statistical analysis, like Principal Component Analysis (PCA) and histogram-based distance, have been

[1] https://aws.amazon.com/elasticloadbalancing/

recently proposed in literature [13–15]. The proposal in [15] is especially suited for this task because it can achieve a VMs classification with an accuracy close to 100% (that is, nearly error free).

Figure 1 illustrates the periodic VMs placement in a cloud data center that adopts the proposed approach. Multiple VMs are grouped into classes (top part of the Figure), with VMs of the same class exhibiting similar resource demands. The VM resource usage is constantly monitored, and the monitoring process may take advantage of the knowledge of VM classes to reduce the amount of data collected and improve scalability, as discussed in [13, 14]. The output of the monitoring process is represented as the data objects marked as "$P_{1...c}$", grouped by class. the samples from the monitoring are fed into a *Prediction* step. This task can be implemented according to multiple techniques, ranging from the simplest solutions assuming that resource demands follow a periodical cycle with a length of 24 hours [18], to complex predictive techniques that can cope with trends, periodic behaviors and state changes [19]. The output of the prediction is an estimation of the resource usage in the future for each class of VMs (data marked with "$F_{1...c}$").

The future demands and the description of the infrastructure of the data center (marked with the letter "I") are the input of the *Consolidation model*, that is the core of our proposal. The consolidation model is based on the bin-packing problem and its output is a solution of such problem that contains the decision (marked with letter "D" in Figure 1) on the mapping between VMs and physical nodes. The placement decision is then applied to the VMs by powering on and off the physical nodes of the cloud infrastructure.

It is worth to note that inaccurate prediction of future resource usages as well as incorrect VMs classification could lead to flawed placement solutions possibly suffering from overload or underload conditions on some physical nodes of the infrastructure. To cope with such condition, dynamic strategies exploiting live VMs migration can be integrated in the system. For example, the solution proposed in [20] of a distributed mechanism based on local information to take decisions on VMs migrations can be easily applied to the depicted cloud data center.

3 Consolidation Models for VM placement

We now discuss the consolidation model, which represents the core of the VMs placement technique. In particular, we start with a description of the existing consolidation models [4, 5] and we discuss the most widely adopted simplifications used to improve the scalability of this task. Next, we present the Class-based placement consolidation model. Finally, we discuss the critical parameters that may affect the model performance of the CBP model.

3.1 Multi-Dimensional Bin Packing model

The consolidation model used for VMs placement is typically based on a multi-dimensional bin packing problem, where one or more VM resources are considered

for consolidation during the next planning period, and the planning period is divided into a set of time intervals **T**. The multi-dimensional bin-packing model is shown in Figure 2. The model input is the prediction of future requirements for every VM in multiple time intervals (the data with the letter "F"). In this case we do not divide the future requirements by class as in Figure 1 because this consolidation model is not class-aware. An additional input of the consolidation model is a description of the data center infrastructure with the available physical nodes and their capacity (the data with the letter "I"). A single bin-packing problem is solved for the whole data center providing the placement of VMs over the nodes of the data center (the output is represented as the data with the letter "D"). The problem can be formalized as follows.

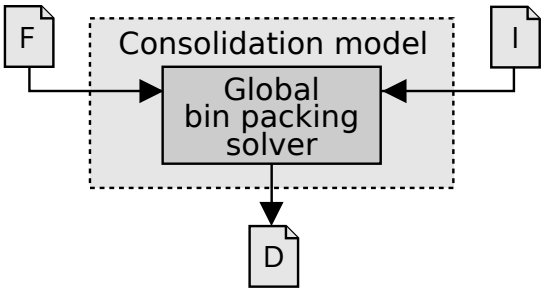


Fig. 2: Consolidation model with global multi-dimensional bin-packing

Let us consider a set **M** of VMs that have to be deployed on a set **N** of physical nodes. The matrix R represents the resource requirements of the VMs over multiple observation time intervals. Although the most general version of the problem involves multiple resources of the VMs, such as CPU, memory, network, and disk [4,21,22], we limit our model to a single resource, the CPU utilization, which is typically a bottleneck resource in Web applications [23] and is often the only considered metric for cloud capacity allocation [4,24]. However, it is worth to note that an extension of our model to include multiple resources (e.g., memory which is the second most common resource considered for VMs placement) is straightforward. Furthermore, the idea of a block-based solution can be extended to several existing multi-resource VMs placement problem formulations, such as the one described in [25]. In our model $R_{m,t}$ represents the CPU requirement of VM m ($m \in \mathbf{M}$) for the time interval t ($t \in \mathbf{T}$). Furthermore, for each node, V models the available capacity on the node: V_n represents the available CPU capacity on node n ($n \in \mathbf{N}$). We can define the optimization problem as follows:

$$\min \sum_{n \in \mathbf{N}} O_n \tag{1}$$

subject to:

$$\sum_{n \in \mathbf{N}} I_{n,m} = 1 \qquad \forall m \in \mathbf{M} \tag{2}$$

$$\sum_{m \in \mathbf{M}} R_{m,t} \cdot I_{n,m} \leq V_n \cdot O_n \qquad \forall n \in \mathbf{N}, \forall t \in \mathbf{T} \tag{3}$$

$$I_{n,m} = \{0,1\} \qquad \forall n \in \mathbf{N}, \forall m \in \mathbf{M} \tag{4}$$

$$O_n = \{0,1\} \qquad \forall n \in \mathbf{N} \tag{5}$$

Where O_n is a binary decision variable that discriminates if a physical node n in the data center is on or off, $I_{n,m}$ is a binary decision variable that decides if VM m is allocated on node n. Expression 1 is the objective function of the optimization problem that aims to minimize the number of used nodes. Due to the set of constraints 2, every VM is allocated exactly on one physical node. The set of constraints 3 expresses the bound that on each node the allocated VMs must not exceed the overall capacity of the node for every considered time interval. Finally, the sets of constraints 4 and 5 model the boolean nature of the decision variables.

When solving bin packing problems, the number of dimensions (in this case the number of time intervals $|\mathbf{T}|$ considered in our problem formulation) has major impact on the time to reach a solution. To improve the scalability of VMs placement problem, a common approach is to reduce the cardinality of constraints 3 in the problem formulation. To this aim, we introduce a different set of time intervals $\mathbf{T}'$ such that $|\mathbf{T}'| < |\mathbf{T}|$ and we bind each new interval $t' \in \mathbf{T}'$ to a set of time intervals $\{t_1, \ldots, t_k\} \in \mathbf{T}$. The new constraint formulation will consider for each VM m a requirement $R_{m,t'} = \max(R_{m,t_1}, \ldots R_{m,t_k})$. In the extreme case, when the number of time intervals is reduced to one, the multi-dimensional bin packing reverts to a one-dimensional bin packing problem. In this case, we can exploit heuristics such as the First Fit Decreasing (FFD) algorithm to reach an approximate solution of the problem in a very short time [12]. However, the reduction of dimensionality typically leads to suboptimal solutions for the VM placement problem.

3.2 Class-Based Consolidation Model

Class-based consolidation exploits the presence of classes of VMs exhibiting similar resource usage. We recall that even if the cloud provider has no direct knowledge of clusters of VMs that host the same software component of the same application, this information can be obtained by the cloud provider by exploiting recently proposed techniques that group together VMs with similar behavior [13–15]. The global bin packing problem, taking into account the whole data center, is reduced to a much smaller problem that takes into account only a few VMs for each class. The reduced size of the problem allows us to solve the multi-dimensional consolidation model with a number of time intervals that would not be possible to consider for the global problem in Section 3.1. The solution of the smaller problem is then replicated as a building block to determine the solution for the global VM placement problem.

Figure 3 shows the Class-based placement. The input consists in the description of the infrastructure (labeled as "I") and of the future VMs requirements, but in this case we assume that the VMs are divided into a set **C** of classes, where all the VMs of a same class present similar resource requirements (data on resource requirements are labeled "$F_{1...c}$" representing the different classes). The basic idea is to divide the global set of VMs in a number $\overline{b}$ of B-blocks all composed by the same number of VMs for each class ($\overline{b}$ is computed by a module of the consolidation model), and one E-block containing the rest of the VMs. These blocks of reduced size are exploited to determine the global solution to the VMs placement problem, as formalized in the rest of this section.

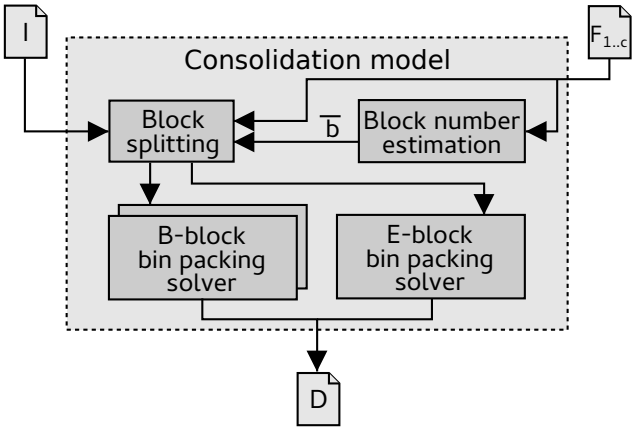


Fig. 3: Class-based consolidation model

For each class $c \in \mathbf{C}$, we define a set of VMs $\mathbf{M_c}$ belonging to class c such that:

$$\bigcup_{c \in \mathbf{C}} \mathbf{M_c} = \mathbf{M}$$

$$\mathbf{M_{c_1}} \cap \mathbf{M_{c_2}} = \emptyset \qquad \forall c_1, c_2 \in \mathbf{C}$$

We recall that VMs belonging to the same class run the same software component of an application, so we can assume that they are characterized by similar resource demand. Hence, we can define their resource demand as:

$$R_{m,t} = R_{c,t} \quad \forall c \in \mathbf{C}, \forall m \in \mathbf{M_c}, \forall t \in \mathbf{T} \tag{6}$$

The global set of VMs is divided in $\overline{b}$ B-blocks composed by the same number of VMs for each class. For each class $c \in \mathbf{C}$, each B-block contains a set $\mathbf{B_c} \subset \mathbf{M_c}$ of VMs belonging to class c. The remaining set of VMs $\mathbf{E_c}$, that are not assigned to any B-block, is assigned to the E-block. Given the number of B-blocks $\overline{b}$, the cardinality of each set is:

$$|\mathbf{B_c}| = \left\lfloor \frac{|\mathbf{M_c}|}{\overline{b}} \right\rfloor \ \forall c \in \mathbf{C} \tag{7}$$

$$|\mathbf{E_c}| = |\mathbf{M_c}| \% \overline{b} \ \forall c \in \mathbf{C} \tag{8}$$

Since all the VMs of a same class exhibit similar resource requirements, the placement solution computed for a single B-block can be replicated on all the remaining B-blocks. The B-block sub-problems is an optimization problem similar to the one in Section 3.1, but applied to the subset of VMs $\bigcup_{c \in \mathbf{C}} \mathbf{B_c}$. Considering the presence of VMs classes (Equation 6), we can express the constraint 3 as:

$$\sum_{c \in \mathbf{C}} \sum_{m \in \mathbf{B_c}} R_{c,t} \cdot I_{n,m} \leq V_n \cdot O_n \quad \forall n \in \mathbf{N}, \forall t \in \mathbf{T}$$

A similar set of constraints applies to the E-block problem.

It is worth to note that, once the B-block problem has been solved, one node for each block may result to be under-utilized. A possible extension of the proposed consolidation model is to include an additional consolidation round including only these nodes. However, in this study, we prefer to discard such a potential improvement focusing more on the scalability gain of the basic Class-based consolidation and on the problem of selecting the size of the B-blocks, even at the cost of being slightly unfair in the evaluation of our own proposal.

To summarize, the global placement solution can be obtained through the following steps: 1) solving the placement problem for one B-block and replicate the solution for all the B-blocks; 2) solving the placement problem for the E-block. The reduced size of these blocks allows us to solve in a reasonable amount of time the corresponding placement problems taking into account a multi-dimensional formulation with multiple time constraints. This approach offers high quality solutions for the global placement problem that can take advantage of complementary workload patterns to minimize the number of required physical nodes.

3.3 Block number estimation

We now focus on how VMs are assigned to the B-blocks and to the E-block. The parameter $\overline{b}$ determines this assignment, hence the selection of the best number of B-blocks is a critical factor for the performance of the proposed CBP technique. The choice of the $\overline{b}$ value is carried out by the Block number estimation component in Figure 3.

The impact of $\overline{b}$ over the consolidation process is twofold. On one hand, as $\overline{b}$ is reduced, the size of the problems in the B-blocks increases. This may have a detrimental effect on the resolvability of the VMs placement problem due to the computational cost of the large optimization problems for the B-blocks. On the other hand, as $\overline{b}$ grows, we tend to have very small problems, where the amount of unused resources of the nodes in each B-block tends to become relevant. In this case we observe a fragmentation effect that may reduce the quality of the solution (the number of physical

nodes used is much higher than the optimum). The identification of the best value of $\overline{b}$ must solve a trade-off between computational cost and solution quality, ensuring that the splitting of the VMs placement problem is feasible.

We consider and compare two different approaches to automatically compute a suitable value of $\overline{b}$. The first approach is a slightly refined version of what has been first proposed by the authors in [16], while the second option is a novel proposal.

The first approach is mostly focused on scalability issues and aims to create as much B-blocks as possible. We define L as the cardinality of the smaller class in the system, that is $L = \min(\{|\mathbf{M_c}|, \forall c \in \mathbf{C}\})$. If we choose $\overline{b} = L$ as in [16], we have as much B-blocks as possible, with the guarantee that at least a representative for each class of VMs is included in the B-block. However, the remaining E-block may be much bigger than the B-block, and we may still encounter scalability problems. Hence, we refine this approach by computing a value $b^{\dagger}$ for $\overline{b}$ using an iterative approach. We start with the maximum value possible of $\overline{b} = L$ and we compute the values of $|\mathbf{B_c}|$ and $|\mathbf{E_c}|$. Referring to the problem in Section 3.2, we recall that Equations 7 and 8 define the number of VMs of a generic class c into a B-block and E-block, respectively. We then evaluate the following constraint:

$$|\mathbf{B_c}| \geq 1 \qquad \forall c \in \mathbf{C} \tag{9}$$

$$\sum_{c \in \mathbf{C}} |\mathbf{B_c}| \geq \sum_{c \in \mathbf{C}} |\mathbf{E_c}| \tag{10}$$

Constraint 9 requires the B-block to contain at least a VM for each class, while constraint 10 is motivated by the need to avoid a block splitting where the B-block remains small and the E-block becomes a huge and intractable problem. If the constraints are satisfied, we return the found value of $\overline{b}$ as $b^{\dagger}$. Otherwise, we decrease $\overline{b}$ and we re-iterate the process. It is intuitive that this approach may increase the risk of sub-optimal global VM placement due to the sub-utilization of at least one physical node for each B-block. However, the small problem size should increase the possibility to solve the B-block and E-blocks problems to optimality (without the need to rely on over-simplified heuristics).

The second approach to determine $\overline{b}$ tries to use large B-blocks as long as this does not determine scalability issues. Also in this case we compute b^{*} (the value of $\overline{b}$ identified by the second approach) through an iterative approach trying to distribute VMs between B-blocks and E-block until a set of constraints is satisfied. Besides the constraints previously defined as 10 and 9, we consider also:

$$\sum_{c \in \mathbf{C}} |\mathbf{B_c}| \leq S \tag{11}$$

Constraint 11 places a maximum size on the VMs in a B-block; the presence of this bound is important because in a previous study [16] we demonstrate that, as the problem size grows, the bin-packing problem becomes unmanageable and cannot be solved. We do not need to place a bound for the maximum size of the E-block because it automatically derives from constraint 10.

The iterative approach starts with an initial value of $\overline{b} = \lceil \frac{|M|}{S} \rceil$. This value derives from constraint 11: a lower value of $\overline{b}$ would automatically violate this condition. If all the constraints are satisfied, we have an acceptable block splitting and we return the found value as b^*. Otherwise, we increment $\overline{b}$ and we re-evaluate the constraints until we find b^*. The maximum possible value for $\overline{b}$ is L: any higher value of $\overline{b}$ would violate the inequality in constraint 9.

4 Experimental results

In this section we present the results of the experimental evaluation of the proposed CBP placement technique. For our evaluation, we consider the quality of the solution and the corresponding resolution time. In the rest of this section we describe the experimental setup and present the results of different experiments aimed at: tuning the S parameter that indicates the B-block maximum size; evaluating the impact of the $\overline{b}$ parameter to identify the best approach to compute the number of B-blocks; comparing the CBP proposal with MBP consolidation models in terms of scalability and solution quality; evaluating the CBP placement performance for workloads with a different composition in term of complementary patterns.

4.1 Testbed description

We obtained an extensive dataset from a private cloud data center. The set contains up to 1200 VMs traces for the resource usage of Web/application/database servers and ERP applications, where the VMs belongs to 44 different classes, with each class containing from 10 to 50 VMs. For our experiments we consider only the CPU resource, as described in the problem formalization of Section 3.1. The CPU traces are the input for the consolidation model in the VM placement problem; the resource usage is measured in intervals of 5 minutes, as in other experiments in literature [26]. We also consider workloads with a different percentage of complementary patterns, ranging from 10% to 90%, to evaluate the sensitivity of the CBP placement performance to the workload composition; when not differently specified, the workload contains a 40% of complementary patterns, that is consistent with cloud workloads considered in other studies [22]. We consider two workloads, described as two time series of CPU requirements $R_{i,t}, R_{j,t}, i, j \in [1, C]$ to be complementary if their sum is roughly 100% throughout the whole considered period. In a more formal way, we can write this condition as $|R_{i,t} + R_{j,t} - 100\%| \leq \varepsilon, \forall t \in \mathbf{T}$, where ε is a tolerance that in our experiment we set to 15%.

For the experimental evaluation, we consider multiple scenarios characterized by different numbers of VMs to be placed on the physical nodes of the virtualized data center. In particular, we consider a VMs set size ranging from 150 to 1200 VMs. For each VM, the CPU utilization is in the range [0%-100%] with an average value of 54%. For each physical node the CPU capacity is 800%, meaning that each node can host 8 VMs with CPU utilization of 100%. For each scenario, we compare different consolidation models operating over a planning period of 24 hours. The proposed

Class-Based Placement (CBP) is solved with 288 five-minutes time intervals. For the Multiple Bin Packing (MBP) model, we consider multiple setups with a different dimensionality of the problem (in terms of number of time constraints). The considered numbers of dimensions for the MBP model are 288 (five-minutes intervals), 24 (1 hour), 2 (12 hours) and a single time interval (24 hours). We have also implemented a First Fit Decreasing (FFD) heuristic [12], that we used as a term of comparison to evaluate the placement solution quality. The FFD heuristic does not allow to consider time constraints, hence it is evaluated with one time interval of 24 hours. All the experiments are run on 2.4 GHz, 16 cores Intel Xeon with 16 GB RAM, using IBM ILOG CPLEX 12.6 as the optimizer solver[2].

As a metric for the VMs placement solution quality, we consider the number of physical nodes that are required for the allocation. The number of nodes for each solution is expressed with respect to an estimation of the optimal solution for the considered scenario. This metric is similar to the competitive ratio [27] used for the on-line bin packing algorithms and measures the *overhead* of the proposed solution against the optimum. Lower values (as close as possible to 100%) means higher quality solutions. The MBP model with five minute time interval (MBP-5min) represents a lower bound for all the feasible allocations, as this consolidation model exploits all the available information to find an optimal solution. However, the number of variables and constraints for this model increases rapidly with the VMs set size, producing an optimization problem instances whose computation takes extremely long times or does not produce any feasible solution due to the huge main memory requirements, that may finally cause the solver to abort the optimization processing. For this reason, we will use the objective function value of the LP relaxation of the MBP-5min consolidation model (in Section 3.1) as a lower bound for the optimal number of physical nodes to use. In other words, we relax the boolean nature of the decision variables, assuming that parts of a VM can be assigned to different physical nodes. In the formulation this corresponds to simply removing the constraint 4. This allocation is obviously not feasible from a technical point of view but can be easily computed, hence we exploit it as a convenient lower bound for any feasible allocation [4].

It is worth to note that for many problems, starting from a medium size (i.e. starting from 250 VMs), the resolution of the MPB consolidation models may take long times, such as hours or days, even for a limited number of time intervals. For that reason, we use a time limit of 30 minutes (1800 seconds) for each problem, and consider the best integer solution found as the solution of the placement problem, as commonly done in similar research studies [4, 28].

4.2 B-block Maximum Size Tuning

This first analysis aims to identify the value of the threshold S that determines the maximum size for the B-block in constraint 11. Basically, we aim to identify the maximum size of the bin packing problem that can be solved to optimality with five minute time intervals within the considered time limit. To this aim, we apply the

[2] www.ibm.com/software/commerce/optimization/cplex-optimizer/

MBP-5min model to scenarios with a VMs set size ranging from 150 to 400 VMs and measure the corresponding resolution times.

Table 1: Resolution times for MBP-5min consolidation model

VMs Set Size	Resolution time (s)
150	233.09
200	270.59
250	1800 (L)
300	1800 (L)
350	No integer solution
400	No integer solution

The results in Table 1 show that, as the number of considered VMs grows from 150 to 200, the time to reach the optimal solution increases of about 16%. When the number of VMs further grows, the solver is no longer able to find an optimal solution due to the time limitation. For scenarios with 250 and 300 VMs, the solver can reach a non optimal but feasible integer solution within the time limit, while for higher VMs set size no integer solution is found. This preliminary analysis provides us with the value for the threshold S used to automatically compute the number of B-blocks b^*: in our experiments we consider $S = 300$ as the size of the largest B-block that can be solved within the time limit.

It is worth to note that the choice of the value for the threshold S has an important consequence: the scenarios with a number of VMs lower than 300 are resolved with just one B-block that includes all the VMs of the system. In this case, the CBP-5min consolidation model is equivalent to the MBP-5min solution. For this reason, in the rest of the paper we only consider scenarios with a number of VMs greater than 300, since only in these medium-large scenarios the difference between proposed and existing placement techniques can be appreciated.

4.3 Impact of $\overline{b}$ parameter

We now aim to evaluate the impact of the $\overline{b}$ parameter, which represents the number of B-blocks, on the performance of the Class-based placement technique. Our goal is to identify which approach provides better results between the two possible alternatives to compute $\overline{b}$ outlined in Section 3.3.

For all the scenarios with a VM set size ranging from 400 to 1200, the first experiment compares the solution quality achieved for the values of $\overline{b}$ computed accordingly to the two approaches, $b^\dagger$ and b^*. Table 2 shows the solution qualities achieved for each VM set size.

The results clearly show that the proposed algorithm for the determination of b^* allows to obtain significant improvements in the solution quality achieved by the CBP model for every considered scenario: the quality of the solutions for $\overline{b} = b^*$ (second column) ranges from 103.2% to 107.7%, while for $\overline{b} = b^\dagger$ (third column) the quality ranges from 106.8% to 115.9%. This result is motivated by the lower number of

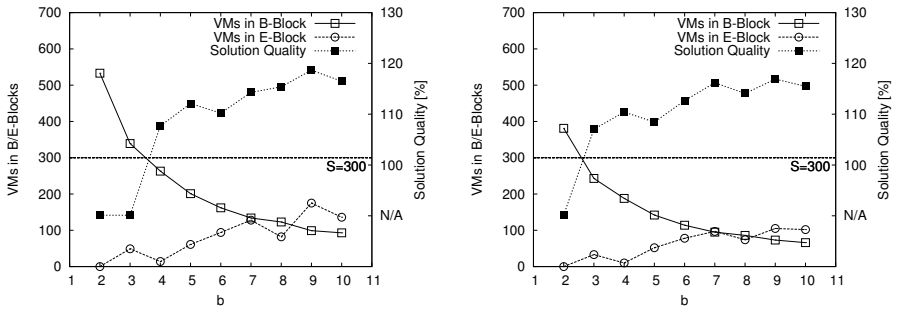
Table 2: Solution quality for CBP-5min model [%]

VMs Set Size	Number of B-blocks ($\overline{b}$)	
	$\overline{b} = b^*$	$\overline{b} = b^\dagger$
400	103.2 ($\overline{b} = 2$)	112.9 ($\overline{b} = 8$)
500	104.5 ($\overline{b} = 2$)	115.9 ($\overline{b} = 8$)
600	105.5 ($\overline{b} = 2$)	111.1 ($\overline{b} = 8$)
700	106.6 ($\overline{b} = 3$)	110.3 ($\overline{b} = 8$)
800	107.0 ($\overline{b} = 3$)	114.1 ($\overline{b} = 10$)
900	105.1 ($\overline{b} = 3$)	108.8 ($\overline{b} = 8$)
1000	104.5 ($\overline{b} = 3$)	106.8 ($\overline{b} = 10$)
1100	107.7 ($\overline{b} = 4$)	114.3 ($\overline{b} = 10$)
1200	105.5 ($\overline{b} = 4$)	113.8 ($\overline{b} = 10$)

B-blocks determined by the approach proposed in this paper: while the value of $b^\dagger$ ranges from 8 to 10, for the same scenarios the values of b^* go from 2 to 4. The high number of B-blocks corresponding to the $b^\dagger$ values increase the risk of wasting of resources due to the sub-utilization of at least one physical node for each B-block. On the other hand, the small number of b^* B-blocks favors high quality solutions for VMs placement.

In the second experiment, for each VMs set size we force the $\overline{b}$ parameter to range from 2 to the maximum allowed value L, which is the cardinality of the smallest VMs class. For each value of $\overline{b}$, we evaluate the number of VMs in B-blocks and E-blocks, and the quality of the solution in terms of VMs placement. We achieve similar results across all the scenarios: for space reasons we do not show all the results, but we discuss the cases of 1100 and 800 VMs as significant examples of large size scenarios.

Figures 4(a) and 4(b) show the results as $\overline{b}$ ranges from 2 to 10 (L) for the scenario with 1100 and 800 VMs, respectively. The graphs also show the line corresponding to the threshold $S = 300$, that we use to define the optimal value of b^*.



(a) Set size: 1100 VMs (b) Set size: 800 VMs

Fig. 4: Block size and solution quality for varying numbers of B-blocks

As expected, in both graphs the number of VMs in the B-blocks decreases as $\overline{b}$ increases. If we observe the values of the solution quality as $\overline{b}$ grows, we note that

both graphs show very clearly the trade-off between resolvability and quality related to the choice of the number of B-blocks. For example, Figure 4(a) shows that for $\overline{b}$ equal to 2 and 3 the solver is not able to reach a feasible integer solution within the 30 minutes limit (solution quality = N/A) because of a too high number of VMs in the B-block. This confirms our assumption that the problem with more than 300 VMs in a block is hardly solvable with constraints of 5-minutes time intervals. We also observe that the best solution is achieved for $\overline{b} = 4$, that corresponds to the value of b^* identified by the algorithm proposed in this paper (Section 3.3): for higher values of $\overline{b}$, the number of required physical nodes slightly grows, leading to a degradation of the solution quality. We recall that the solution quality is measured as the number of physical nodes required for the allocation with respect to the objective function value of the LP relaxation of MBP (5-min); in other words, a solution quality of 110% means that the allocation required 10% physical nodes more with respect to the solution of the LP relaxation problem. The latter result confirms that increasing the number of B-blocks is likely to cause the risk of sub-optimal global VM placement due to the waste of resources in at least one physical node of each B-block.

Similar observations are valid for Figure 4(b), that represents the scenario with 800 VMs. It is important to note that also for this scenario, as for all the VMs set sizes not reported here, the value b^* corresponds to the number of B-blocks the leads to the best quality solution for the VM placement problem. This experiment confirms that the approach proposed in this paper to automatically determine the number of B-blocks is effective in addressing the trade-off between resolvability and solution quality.

4.4 Scalability analysis

In this experiment we compare different consolidation models to evaluate if they can reach an optimal or feasible solution within the expected time limit of 30 minutes. Table 3 shows for which scenarios it was possible to solve the problem instances to optimality (S), reach an integer solution even if not optimal (L), or not even find any feasible integer solution (N) within the time limit. We evidence the cells related to unsolvable problem instances with a gray background. For the CBP technique we consider both the approaches to compute the number of B-blocks: b^* and $b^\dagger$ (second and third column of the table, respectively). Moreover, we report two values separated by a slash symbol: the first refers to the resolution of the B-block and the second to the E-block.

We observe that, for MBP models considering short time intervals of 5 minutes and 1 hour, it is not possible to find a feasible integer solution within the time limit starting from medium sized problems of 400 and 600 VMs, respectively; for larger time of 12 hours and 1 day, the size of resolvable problems grows to 1000 and 1100, respectively. On the other hand, the breakdown in building blocks allows the CBP model to find a feasible integer solution for every VMs set size, with the possibility to solve to optimality even scenarios up to 600 VMs. The results confirm that the CBP technique allows us to solve significantly larger problems with respect to a MBP approach, even when the MBP problem considers few or just one time intervals. If

Table 3: Scalability evaluation of MBP/CBP consolidation models

VMs Set Size	Consolidation Models					
	CBP (b^*)	CBP ($b^\dagger$)	MBP 1d	MBP 12h	MBP 1h	MBP 5min
400	S/S	S/S	L	L	L	N
500	S/S	S/S	L	L	L	N
600	S/S	S/S	L	L	N	N
700	L/S	S/S	L	L	N	N
800	L/S	S/S	L	L	N	N
900	L/S	L/S	L	L	N	N
1000	L/S	L/S	L	L	N	N
1100	L/L	L/S	L	N	N	N
1200	L/L	L/S	N	N	N	N

we observe the second and third columns of the table, we note that the choice of the number of B-blocks of the CBP model can affect the scalability of the placement. For $\overline{b}$ equal to b^* the number of B-blocks tends to be low, thus generating blocks with a higher number of VMs with respect to the case of $b^\dagger$. Indeed, the solution with b^* can not solve to optimality scenarios with more than 600 VMs. However, it is important to note that the solver can always reach a feasible integer solution even for the larger scenarios; this is due to the threshold imposed on the maximum size of the B-block.

4.5 Solution quality comparison

Let us now evaluate the quality of the solutions achieved by the CBP proposal with respect to the MBP consolidation model for the different scenarios. Fig. 5 shows the solution qualities of the consolidation models for VM set sizes ranging from 400 to 1200. The CBP model considered for this experiment exploits a number of B-blocks equal to b^*. This graph does not report the results for the MBP-5min model, because it cannot find any feasible solution for instances larger than 300 VMs, as discussed in Section 4.2.

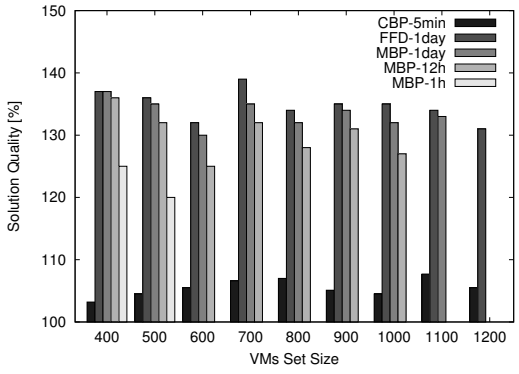


Fig. 5: Solution quality for different consolidation models

The graph clearly shows how the increase in the number of VMs causes the impossibility to find a feasible integer solution within the time limit for the MBP models: we see the histogram bars referred to MBP models gradually disappearing as the number of VMs increases. For 1200 VMs, only the FFD heuristic and the proposed CBP model are able to find a feasible solution within 30 minutes. From the graph is also evident how MBP models with few time constraints support the resolution of problem instances with larger numbers of VMs, but at the expense of solution quality.

The results for scenarios with 400 and 500 VMs show that CBP-5min significantly outperforms the MBP-1h model, showing that for medium sized scenarios an hourly aggregation of the resource usage data does not provide the consolidation problem with enough information to find a solution as efficient as our proposal. For scenarios with more than 500 VMs, the difference of quality between the solutions provided by CBP-5min and MBP models further increases: the difference with respect to the best option between MBP-12h and MBP-1d models ranges from 19% to 26% .

We can conclude that the CBP-5min allows to find a feasible solution for every VMs set size, up to 1200 VMs. While for small scenarios (less than 300 VMs) the CBP is equivalent to a MBP-5min that could solve to optimality the global allocation problem, starting from medium-sized scenarios (400 VMs) the proposed solution outperforms other consolidation models both in terms of resolution time and number of required physical nodes.

4.6 Sensitivity to workload composition

The adoption of the CBP technique makes it feasible to resolve large placement problems with multiple time constraints, thus allowing to leverage the presence of complementary workload patterns to minimize the number of physical nodes. In this last experiment we evaluate the sensitivity of the CBP proposal to the workload composition, considering a percentage of complementary patterns ranging from 10% to 90% in two scenarios with 600 and 1000 VMs. Figure 6 shows the solution quality gain, that is the difference in the solution quality between the CBP-5min placement technique and the MBP-1d model, which does not consider multiple time intervals.

As expected, the solution quality gain increases as the percentage of complementary workload patterns increases, confirming the capability of the CBP proposal to take advantage of complementary resource utilization to achieve a good VMs placement on the physical infrastructure. Moreover, it is interesting to note that, even for low percentages of complementary patterns (10%), the CBP placement significantly outperforms the MBP-1d model. This result shows that solving a placement problem with a fine granularity in terms of time intervals (5 minutes) allows the system to exploit any existing difference in the VMs resource utilization patterns to minimize the number of used physical nodes. This is an important results because it confirms that the CBP solution represents a winning option for a wide range of workloads and not only in presence of complementary patterns.

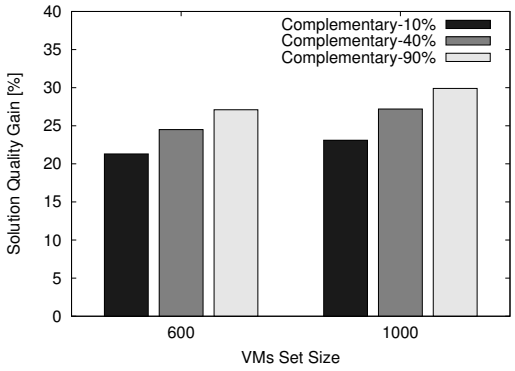


Fig. 6: Solution quality gain of CBP-5min over MBP-1d consolidation model

5 Related work

The placement of VMs over the physical nodes of cloud data centers represents a critical task to limit the costs of the infrastructure management and avoid waste of computing resources. Efficient VM placement can reduce both the environmental footprint of IT infrastructures and the energy-related cost for the cloud business firms [3, 2].

The main goal of VMs placement is to minimize the number of physical nodes required to allocate a given set of VMs in the data center. To this purpose, large data centers can leverage techniques such as selectively powering down idle servers or using hardware support for idle sleep states [9]. However, exploiting these techniques requires the resolution of the optimization problem described in Section 3, to determine how to map VMs over the physical nodes of the cloud infrastructure. This problem is a multi-dimensional bin-packing with bounds related to the requirement of multiple VM resources at different time intervals over a future planning period. Solving this problem is a challenge from a computational point of view, where standard optimization algorithms struggle to reach an optimal solution within acceptable time frames. To reduce the dimensionality of the problem, three classes of solutions have been proposed in literature or applied in real systems: use of VMs nominal maximum requirements, problem simplification discarding several dimensions of the problem, and adoption of heuristics.

Considering only the nominal maximum requirements of each VM is the most straightforward solution. Basically, we simply discard any information about VM demands over time and we consider only the nominal VM peack requirements. This solution is widely adopted when PaaS-level (often threshold-based) are applied to automatically spawn or destroy VMs based on resource utilization and service level measurements [29]. As this approach simplifies the bin-packing problem to the point where no actual problem solver is required (the VMs may be dimensioned to be an exact fraction of the node computational power), it is widely applied [6–8]. However, such solution introduces the unreal assumption that every VM uses the 100% of its resources. Any under-utilized VM determines a waste of resources in the data center

and increases the carbon footprint of the cloud infrastructure. The PaaS-level creation or dismissal of VMs may mitigate the waste of resources, but it requires an additional level of software installed within the VMs. Our approach has a wider applicability because we focus on a IaaS-only vision of the data center management.

The second approach reduces the problem dimensionality by limiting the number of VM resources and time intervals that are considered in the bin-packing problem. For example, instead of considering multiple resources (CPU, memory, network I/O, disk I/O) and a fine-grained division of the planning period, the focus may be limited to just the CPU requirement during a 24-hour long time interval [10,4]. Our experiments clearly shows the high penalty in terms of sub-optimal VM placement that is caused by reducing the number of time periods considered in the VM placement problem and we show how our proposal outperforms these solutions. It is worth to note that, although we consider only CPU requirements in our experiments, our approach can be easily extended to take into account additional resources. Furthermore, the basic idea of splitting the global bin packing problem into smaller building blocks can be applied to other formalizations of the VMs placement problem, such as the one in [25].

Finally, the third approach to address the computational issues of the bin-packing problem is to exploit heuristics to reduce the computational cost of the solution. However, as pointed out in [11], most research is focused on problems with few dimensions, while if we consider the impact of multiple resources considered in multiple time intervals in a future planning period, the number of dimensions significantly grows to the order of hundreds. The most popular heuristics are applied to problems characterized by a number of dimensions ranging from one to three [30,31]. As the dimensionality of the problem exceeds these values, the quality of the solution identified by the heuristics significantly drifts away from the optimum. Our approach is completely different from these studies, because we reduce the number of VMs and nodes involved in the bin-packing problem to form a building block of limited size where we can easily apply complex optimization, without the need to reduce the number of dimensions and constraints.

A first version of a class-based placement technique was proposed by the authors in [16]. However, that preliminary work left as an open issue the critical choice of the number of B-blocks. In this paper we propose a technique to automatically determine that number and evaluate the performance of our proposal under a wide range of scenarios. Moreover, we carry out a sensitivity analysis of the class-based placement performance with respect to different workload compositions.

6 Conclusions

In this paper we tackle the critical problem of VMs placement in IaaS cloud computing data centers, with particular attention to the scalability challenges of this task in large cloud infrastructures. To cope with the scalability issues of current placement techniques, we propose an alternative approach where VMs are not considered as black boxes with independent resource requirements. Exploiting recent solutions that can cluster together VMs exhibiting similar behaviors in terms of resource usage,

we propose a novel technique, namely Class-Based Placement (CBP), that solves a small-size placement problem and replicates it as a building block to obtain the global solution. The number of building blocks used to split the global problem is automatically determined by the proposed CBP technique.

An extensive set of experiments demonstrates that our proposal outperforms existing solutions both in terms of scalability and quality solution, achieving a higher performance gain for large data centers, which represent the most challenging scenario for cloud computing. In particular, our proposal may reduce from 19% to 26% the number of physical nodes required to host the VMs with respect to widely used alternatives in case of medium-large scenarios. Finally, we evaluate the sensitivity of the CBP placement performance with respect to the percentage of complementary patterns in the cloud workload to give insights on the benefits achievable through our proposal in case of different workload compositions.

References

1. J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the Future*, vol. 2007, pp. 1–16, 2012.
2. EPA, "Data center consolidation plan," US Environmental Protection Agency, Tech. Rep., 2011.
3. Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.
4. T. Setzer and M. Bichler, "Using matrix approximation for high-dimensional discrete optimization problems: Server consolidation based on cyclic time-series data," *European Journal of Operational Research*, vol. 227, no. 1, pp. 62–75, 2013.
5. B. Speitkamp and M. Bichler, "A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers," *Services Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 266–278, 2010.
6. B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman *et al.*, "Reservoir – when one cloud is not enough," *IEEE computer*, vol. 44, no. 3, pp. 44–51, 2011.
7. K. Mills, J. Filliben, and C. Dabrowski, "Comparing VM-Placement Algorithms for On-demand Clouds," in *Proc. of IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom)*, Athens, Greece, Nov. 2011.
8. C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A Scalable Application Placement Controller for Enterprise Data Centers," in *Proc. of the 16th International Conference on World Wide Web (WWW)*, Banff, Alberta, Canada, May 2007.
9. L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE computer*, vol. 40, no. 12, pp. 33–37, 2007.
10. T. Setzer and A. Stage, "Decision support for virtual machine reassignments in enterprise data centers," in *Proc. of Network Operations and Management Symposium*, Osaka, Japan, Apr. 2010.
11. G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, 2007.
12. M. Kao, *Encyclopedia of Algorithms*. Springer-Verlag New York, Inc., 2008.
13. C. Canali and R. Lancellotti, "Automatic Virtual Machine Clustering based on Bhattacharyya Distance for Multi-cloud Systems," in *Proc. of International Workshop on Multi-cloud Applications and Federated Clouds (MultiCloud*, Prague, Czech Republic, Apr. 2013.
14. ——, "Improving Scalability of Cloud Monitoring Through PCA-Based Clustering of Virtual Machines," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 38–52, 2014.
15. ——, "An Adaptive Technique to Model Virtual Machine Behavior for Scalable Cloud Monitoring," in *Proc. of IEEE Symposium on Computers and Communications (ISCC)*, Madeira, Portugal, Jun. 2014.
16. ——, "Exploiting Classes of Virtual Machines for Scalable IaaS Cloud Management," in *Proc. of the 4th Symposium on Network Cloud Computing and Applications (NCCA)*, Munich, Germany, Jun. 2015.

17. M. Rabinovich and O. Spatscheck, *Web caching and replication*. Boston, MA, USA: Addison-Wesley Boston, USA, 2002.
18. A. K. Iyengar, M. S. Squillante, and L. Zhang, "Analysis and characterization of large-scale Web server access patterns and performance," *World Wide Web*, vol. 2, no. 1-2, pp. 85–100, 1999.
19. S. Casolari and M. Colajanni, "On the selection of models for runtime prediction of system resources," in *Run-time Models for Self-managing Systems and Applications*, ser. Autonomic Systems, D. Ardagna and L. Zhang, Eds. Springer, 2010, pp. 25–44.
20. C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic Consolidation of Virtual Machines in Self-Organizing Cloud Data Centers," *Cloud Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 215–228, July 2013.
21. W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179 – 196, 2013.
22. R. Zhang, R. Routray, D. M. Eyers, D. Chambliss, P. Sarkar, D. Willcocks, and P. Pietzuch, "IO Tetris: Deep storage consolidation for the cloud via fine-grained workload analysis," in *Proc. of 4th IEEE International Conference on Cloud Computing, (CLOUD)*, Washington, DC, USA, Jul. 2011.
23. M. Andreolini, S. Casolari, and M. Colajanni, "Models and framework for supporting runtime decisions in Web-based systems," *ACM Transactions on the Web*, vol. 2, no. 3, pp. 1–43, 2008.
24. B. Addis, D. Ardagna, B. Panicucci, M. Squillante, and L. Zhang, "A hierarchical approach for the resource management of very large cloud platforms," *Dependable and Secure Computing, IEEE Transactions on*, vol. 10, no. 5, pp. 253–272, 2013.
25. M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and distributed Computing*, vol. 70, no. 9, pp. 962–974, 2010.
26. B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, and L. Zhang, "A hierarchical approach for the resource management of very large cloud platforms," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 253–272, 2013.
27. L. Epstein, L. Favrholdt, and J. Kohrt, "Comparing online algorithms for bin packing problems," *Journal of Scheduling*, vol. 15, no. 1, pp. 13–21, 2012.
28. L. Zhang and D. Ardagna, "SLA Based Profit Optimization in Autonomic Computing Systems," in *Proc. of the 2nd International Conference on Service Oriented Computing (ICSOC)*, New York, NY, USA, Nov. 2004.
29. X. Zhu, D. Young, B. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D.Gmach, R. Gardner, T. Christian, and L. Cherkasova:, "1000 islands: An integrated approach to resource management for virtualized data centers," *Journal of Cluster Computing*, vol. 12, no. 1, pp. 45–57, 2009.
30. O. Faroe, D. Pisinger, and M. Zachariasen, "Guided local search for the three-dimensional bin-packing problem," *Informs journal on computing*, vol. 15, no. 3, pp. 267–283, 2003.
31. T. G. Crainic, G. Perboli, and R. Tadei, "Ts 2 pack: A two-level tabu search for the three-dimensional bin packing problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 744–760, 2009.