Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms

ZOLTÁN ÁDÁM MANN, Budapest University of Technology and Economics

Data centers in public, private, and hybrid cloud settings make it possible to provision virtual machines (VMs) with unprecedented flexibility. However, purchasing, operating, and maintaining the underlying physical resources incurs significant monetary costs and environmental impact. Therefore, cloud providers must optimize the use of physical resources by a careful allocation of VMs to hosts, continuously balancing between the conflicting requirements on performance and operational costs. In recent years, several algorithms have been proposed for this important optimization problem. Unfortunately, the proposed approaches are hardly comparable because of subtle differences in the used problem models. This article surveys the used problem formulations and optimization algorithms, highlighting their strengths and limitations, and pointing out areas that need further research.

Categories and Subject Descriptors: D.4.7 [**Operating Systems**]: Organization and Design—*Distributed systems*

General Terms: Algorithms, Design, Management, Performance

Additional Key Words and Phrases: Cloud computing, data center, virtual machine, live migration, VM placement, VM consolidation, green computing

ACM Reference Format:
Zoltán Ádám Mann. 2015. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. ACM Comput. Surv. 48, 1, Article 11 (August 2015), 34 pages.
DOI: http://dx.doi.org/10.1145/2797211

1. INTRODUCTION

In recent years, the increasing adoption of cloud computing has transformed the IT industry [Buyya et al. 2009]. From a user's perspective, the practically unlimited scalability, the avoidance of up-front investments, and usage-based payment schemes make cloud computing a very attractive option. Besides globally available public cloud solutions, enterprises also take advantage of similar solutions in the form of private clouds and hybrid clouds.

Large, virtualized data centers (DCs) are serving the ever-growing demand for computation, storage, and networking. The efficient operation of DCs is increasingly important and complex [Barroso et al. 2013]. Besides traditional cost factors such as equipment and staff, energy consumption is playing an increasing role because of its

This work was partially supported by the Hungarian Scientific Research Fund (grant OTKA 108947) and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.
Author's address: Z. Á. Mann, Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Magyar tudósok körútja 2, 1117 Budapest, Hungary; email: zoltan.mann@gmail.com.

DOI: http://dx.doi.org/10.1145/2797211

costs and environmental impact. According to a recent study, DC energy consumption is the fastest-growing part of the energy consumption of the ICT ecosystem; moreover, the initial cost of purchasing the equipment for a DC is already outweighed by the cost of its ongoing electricity consumption [Digital Power Group 2013].

Cloud DCs typically make extensive use of virtualization technology to ensure isolation of applications while at the same time allowing a healthy utilization of physical resources. Virtual machines (VMs) are either provided directly to the customers in case of an Infrastructure-as-a-Service (IaaS) provider or used to wrap the provisioned applications in case of Software-as-a-Service (SaaS) or Platform-as-a-Service (PaaS) providers [Zhang et al. 2010].

An attractive option for saving energy in DCs is to *consolidate* the VMs to the minimal number of physical hosts and switching the unused hosts off or at least to a less power-hungry mode of operation (e.g., sleep mode). However, VM consolidation that is too aggressive can lead to overloaded hosts with negative effects on the delivered quality of service (QoS), thus potentially violating the service level agreements (SLAs) with the customers. Hence, VM allocation must find the optimal balance between QoS and energy consumption [Buyya et al. 2010; Srikantaiah et al. 2009].

Good VM allocation also helps to serve as many customer requests as possible with the given set of resources, thus amortizing the expenses related to purchasing, operations, and maintenance of the equipment (computing, network, and storage elements, as well as the physical DC infrastructure with cooling, redundant power supplies, etc.). In fact, achieving good utilization of server capacities was one of the key drivers behind the wide spread of virtualization technology. Today, virtualization and the live migration of VMs between hosts are key enablers of efficient resource allocation in DCs [Beloglazov and Buyya 2010b].

Besides using its own DC, a cloud provider (CP) can—in times of extremely high demand—use VMs from other providers as well, such as in a cloud federation or hybrid cloud setting [Casalicchio et al. 2013]. This way, the CP can serve its customers without restrictions. However, this further enlarges the search space for the best allocation.

In this article, we focus on the VM allocation problem, i.e., determining the placement of VMs on physical hosts or using external providers, taking into account the QoS guarantees, the costs associated with using the hosts—with special emphasis on energy consumption—and the penalties resulting from VM migrations. Several algorithms have been proposed in the literature for this important and challenging optimization problem. However, these algorithms address slightly different versions of the problem, differing for example in the way the communication between hosts is modeled or how multicore CPUs are handled. Lacking a generally accepted definition of the VM allocation problem, or some versions of the problem, many researchers came up with many different versions, and these differences can have substantial impact on algorithm runtime and/or on the applicability of the algorithm. This somewhat chaotic situation is even worsened by the fact that some authors failed to explicitly and precisely define the version of the problem that they are addressing, so this must be figured out indirectly from the algorithms that they proposed or the way in which they evaluated their algorithms.

The primary aim of this work is to "tidy up" the relevant problem formulations. Specifically, in Section 2, we start with a discussion of the context and the actors of the VM allocation problem, followed by a description of the characteristics of the problem in Section 3. Section 4 presents a survey of the problem formulations existing in the literature, showing how those works fit into our general framework. Although our main focus is on problem formulations, we complete the survey of the literature with a brief description of the algorithms that have been proposed and how they were evaluated in Section 5. This is followed by a more detailed description of the most important

algorithmic works of the field in Section 6, a discussion of the areas that we believe will need further research in Section 7, and our concluding remarks in Section 8.

Here, we are mostly concerned with the details of *problem formulations* and their *algorithmic implications*. Technical details relating to infrastructure, architecture, and implementation issues are covered only as necessary for the aim of this work.

2. PROBLEM CONTEXT

The VM allocation problem is one of the core challenges of using the cloud computing paradigm efficiently. Cloud computing encompasses several different setups, and depending on this, the VM allocation problem also has different flavors.

Usually, cloud computing scenarios are classified along two dimensions [Zhang et al. 2010; Strauch et al. 2011]. One dimension concerns the nature of the offered service, differentiating between three categories: IaaS, PaaS, and SaaS. The other dimension refers to whether the service is provisioned in-house (private cloud), by a public provider (public cloud), or a combination of the two (hybrid cloud). The three possibilities along both dimensions give rise to nine different possibilities.

Another classification focuses on service deployment scenarios [Li et al. 2012]. Here it is assumed that a service provider (SP) would like to deploy a service on the infrastructure provided by one or more infrastructure providers (IPs) [Talwar et al. 2005]. Depending on the relationship(s) between the SP and IP(s), the following scenarios are distinguished [Li et al. 2012]:

—*Public cloud*: The SP makes use of the IP's infrastructure offering available to the general public.
—*Private cloud*: The SP uses its own resources so that it also acts as IP.
—*Bursted cloud*: A hybrid of the public and private clouds, in which both in-house resources and resources rented from a public IP are used.
—*Federated cloud*: The SP contracts only one IP, but the IP collaborates with other IPs to share the load in a manner that is transparent to the SP.
—*Multicloud*: The SP uses multiple IPs to deploy (parts of) the service.
—*Cloud broker*: The SP contracts a single broker, which contracts multiple IPs but hides the complexity of the multicloud setup from the SP.

From our point of view, the crucial observation is that in each scenario, there is a need to optimize the allocation of VMs to physical resources, but this optimization may be performed by different actors and may have different characteristics, depending on the exact setup [Rochwerger et al. 2009; Ferrer et al. 2012]. Using the classification of Li et al., the VM allocation problem occurs in the respective scenarios as follows:

—*Public cloud*: The IP must optimize the utilization of its resources to find the best balance between the conflicting requirements on profitability, performance, dependability, and environmental impact.
—*Private cloud*: The same kind of optimization problem occurs for the provider that acts as both SP and IP in this case.[1]
—*Bursted cloud*: Two slightly different optimization problems occur:
  —The IP must solve the same kind of optimization problem as above.
  —The SP must solve a similar problem for its own resources, extended by the possibility to off-load some VMs to an external IP.

[1]However, there can be subtle differences—for example, SLAs tend to be less formal and VM sizes are more flexible.

—*Federated cloud*: The IPs must solve an optimization problem similar to the one the SP faces in the bursted cloud setup (i.e., optimization of own resources coupled with workload off-loading decisions).
—*Multicloud*: Again, two different optimization problems occur:
  —The IPs must solve the same kind of optimization problem as in the public cloud setup.
  —The SP must solve an optimization problem in which the optimal allocation of parts of the service to the IPs is decided.
—*Cloud broker*: From an optimization point of view, this is the same as the multicloud scenario, with the broker taking the role of the SP.

In the following, we try to describe the VM allocation problem in a manner that is general enough to cover the preceding variants, and we make the differences explicit only when necessary. We use the term *cloud provider* (CP) to refer to the entity who must carry out the VM allocation (which can be either the SP or the IP, depending on the setup). We assume that the CP must allocate VMs to a set of available resources. In general, there can be two kinds of resources: they can belong either directly to the CP or the CP can also rent resources from *external cloud providers* (eCPs). Depending on the specific setup, it is possible that the CP has only its own resources and there are no eCPs, but it is also possible that the CP does not own resources and can only select from eCPs. The case in which both internal resources and eCPs are available can be seen as the common generalization of all preceding scenarios.

3. PROBLEM CHARACTERISTICS

Depending on the exact setup, there can be some differences in the most appropriate problem formulation, but in most cases, the main characteristics of the VM allocation problem are the following [Mann 2015b]:

—The CP accommodates VMs on the available physical machines (PMs) or by renting capacity from eCPs.
—The number of VMs changes over time as a result of upcoming requests to create additional VMs or to remove existing VMs.
—The resource requirements (e.g., computational power, memory, storage, network communication) of a VM can vary over time.
—The PMs have given capacity in terms of these resources.
—The use of resources incurs monetary costs and consumes electric power. The magnitude of the costs and power consumption may depend on the type, state, and utilization of the resources.
—VMs can be migrated from one PM to another by means of live migration. This takes some time and creates additional load for the involved PMs and the network.
—PMs that are not used by any VM can be switched to a low-energy state.
—If the QoS requirements of the customer are not met, this may result in a penalty.

In the following, we investigate these aspects in more detail.

3.1. Virtual Machines

A VM is usually characterized by the following:

—The number of CPU cores
—Required CPU capacity per core (e.g., in million instructions per second (MIPS))
—Required RAM size (e.g., in gigabytes)
—Required disk size (e.g., in gigabytes).

Additionally, there can be requirements concerning the communication (bandwidth, latency) between pairs of VMs or a VM and the customer.

All of a VM's resource requirements can vary over time. Depending on the type of application(s) running on the VM, the VM's resource requirements can be relatively stable, changing periodically (e.g., in daily rhythm), or oscillating chaotically. To optimize resource usage, the CP must be well aware of the current resource requirements of the VMs and, even more importantly, the resource requirements expected for the near future [Guenter et al. 2011].

In a public cloud setting, it is common that the CP offers standardized types of VMs. In a private cloud setting, customers usually have more freedom in specifying the parameters of a requested VM.

3.2. Resources

The resources available to the CP can be of two types:

—PMs, owned by the CP
—eCPs, from which VMs can be leased.

These two resource types are significantly different. The CP's own PMs are *white-box* resources: the CP has detailed information about their state (e.g., power consumption characteristics, current workload, temperature), and it is the CP's responsibility to optimize the use of these resources. On the other hand, eCPs represent *black-box* resource pools: the CP has no knowledge about the underlying physical infrastructure; it only knows the interface to request and manage VMs. Obviously, the CP has no direct influence on the underlying physical resources in this case.

Another important difference is that utilizing VMs from eCPs incurs direct costs that are normally higher than using the CP's own resources, as they also cover the eCP's profit. Therefore, a CP will usually first try to use its own resources and use eCPs only as an extension in times of demand peaks. It is also possible that a CP has no resources on its own and uses eCPs only [Genez et al. 2012].

Own PMs can reside in one or more DCs. If two PMs reside in different DCs, this usually leads to higher latencies in the communication between them, compared to the case when both PMs are in the same DC. In addition, live migration is usually done only within DC boundaries.

3.3. PM Characteristics

The *utilization* or *load* of a PM measures to what extent its resources are utilized by the VMs residing on it. The most critical resource in terms of utilization is the CPU. On the one hand, it is the CP's interest to achieve high CPU utilization to make the best use of the available resources. On the other hand, if CPU load is too high, this makes it likely that the VMs residing on the given PM do not receive the required capacity, which may lead to SLA violations and damage customer satisfaction. Too high of a CPU load may also lead to overheating and can accelerate aging of the hardware. For these reasons, many researchers concentrated on CPU load.

However, other resources like memory or disk space can also become a bottleneck [Tomás and Tordsson 2014]. Of particular interest is the cache, because current virtualization technologies do not ensure isolation of the cache usage of individual VMs accommodated by the same PM, leading to contention between them [Koller et al. 2011; Verma et al. 2008b]. Thus, it is important to model and predict the performance interference that can be expected when co-locating a pair of VMs [Kim et al. 2013].

Power consumption of a PM is a monotonously increasing function of the CPU load [Khosravi et al. 2013]. Determining the exact dependence of power consumption on

CPU load is a nontrivial problem on its own [Mobius et al. 2014] and is even application dependent [Koller et al. 2010]. Additionally, the load of other system components (e.g., disk) may also play an important role. However, a linear approximation of power consumption as a function of CPU load works quite well across a wide range of applications and platforms [Rivoire et al. 2008]. Hence, several authors assumed linear dependence on CPU load [Beloglazov et al. 2012; Jung et al. 2010; Gao et al. 2013; Lago et al. 2011; Gmach et al. 2008; Svärd et al. 2014].

The amount of energy actually consumed by a PM not only depends on power efficiency but also on the duration. As shown by Srikantaiah et al. [2009], consolidating an increased amount of workload on a server improves energy consumption up to a certain point, when the usage of some resource of the server starts to saturate. Further increasing the load of the server leads to a slowdown of the execution of applications; since jobs take longer, the energy per job starts to increase [Srikantaiah et al. 2009].

Energy consumption of a server has a substantial static component that does not depend on the load of the server: even if a PM is "empty" (i.e., it accommodates no VM), its energy consumption is nonnegligible. To save more energy, it is therefore necessary to switch empty PMs to a *low-energy state*. In the simplest case, a PM has two states: On and Off. More sophisticated models include multiple states with different characteristics, such as On, Sleep, Hibernate, and Off. Realistically, switching between states takes some time, the amount of which depends on the source and target states. For instance, switching between On and Sleep is usually much quicker than switching between On and Hibernate; however, Hibernate will consume less energy than Sleep [Guenter et al. 2011]. Nevertheless, most of the existing works use only a simplified two-state model.

To react to variations in the utilization, PMs usually offer—either directly or through the virtualization platform—several possibilities. Dynamic voltage and frequency scaling (DVFS) is widely used to scale up or down the frequency of the CPU: in times of high load, the frequency is scaled up to increase performance at the cost of higher power consumption, whereas in times of low load, it is scaled down to decrease power consumption [Lago et al. 2011]. Using virtualization, it is possible to explicitly size the VMs by defining their share of the physical resources, and VMs can also be resized dynamically [Cherkasova et al. 2007]. Scaling requests from the VMs can be used by the virtualization layer to determine the necessary physical scaling [Nathuji and Schwan 2007].

3.4. eCP Characteristics

eCPs may offer VMs in two possible ways: either the eCP predefined some VM configurations from which customers can choose (e.g., Amazon EC2) or customers can define their own VM configuration by specifying the needed amount from each resource (e.g., IC Cloud); this can make a difference in the achievable efficiency [He et al. 2012].

There can be considerable differences between eCPs concerning prices and pricing schemes, and even the same eCP may offer multiple pricing schemes [Li et al. 2013]. For example, some providers offer discounted long-term rental rates and higher rates for the pay-as-you-go model [Genez et al. 2012]. The latter is often based on time quanta like hours. Further, there may be a fee proportional to the use of some resources like network bandwidth [Lampe et al. 2012]. In recent years, a further pricing scheme emerged: spot instances, the price of which depends on the current load of the provider. When the provider has a lot of free capacity, spot instances are cheap, but they become more expensive when the load of the provider is getting higher. Consumers can specify until what price they would like to keep the spot instance [Chohan et al. 2011].

3.5. Communication and Networking

VMs are used by customers to perform certain tasks, which are often parts of a bigger application, such as tiers of a multitier application [Jung et al. 2010]. This results in communication between the VMs. In some cases, this can mean the transfer of huge amounts of data, which may lead to an unacceptable increase in latency or response time as well as increased energy consumption in the affected hardware elements (PMs, routers, switches, etc.).

For the preceding reasons, it is beneficial to place VMs that communicate intensively with each other on the same PM, or at least within the same DC [Beloglazov and Buyya 2010b]. On the other hand, VMs that belong to the same application may exhibit correlation between their loads, increasing the probability that they will peak at the same time; this also has to be considered carefully in the VM allocation [Verma et al. 2009].

In some cases, the available network bandwidth can become a bottleneck [Divakaran et al. 2014]. Some authors model network bandwidth the same way as any other resource of the PMs [Biran et al. 2012; Chaisiri et al. 2009; Oliveira et al. 2012; Rodero et al. 2012; Wood et al. 2009]. Others focus specifically on the communication among the VMs and try to minimize the resulting communication cost [Meng et al. 2010] or makespan [Batista et al. 2007]. Some works use a detailed network model with one or more layers of switches and communication links among switches and between switches and PMs, based on different topologies [Jiang et al. 2012; Biran et al. 2012]. Analogous problems arise also concerning the communication between multiple clouds [Bittencourt et al. 2012a].

A strongly related issue is the mapping of data on storage nodes. Some applications use huge amounts of data that are to be mapped on specialized storage nodes, leading to considerable network traffic between compute nodes and storage nodes. In such cases, the placement of VMs on compute nodes and the placement of application data on storage nodes are two interrelated problems that must be considered together to avoid unnecessarily high network loads [Korupolu et al. 2009].

Besides communication among VMs and between VMs and storage nodes, there is communication with entities outside the cloud. An important example are the users. In several applications, the response time experienced by users is critical. The response time is the sum of the network round trip time and the processing time, and can thus be optimized by serving user requests from a DC offering a combination of low latency to the respective user and quick processing [Keller and Karl 2014].

3.6. Service Level Agreements

By SLA, we mean any agreement between the CP and its customers on the expected service quality. The SLA defines service level objectives (SLOs) [Xiong et al. 2015]: key measures to determine the appropriateness of the service (e.g., availability or response time). The SLA can be a formal document, specifying exactly for each SLO the performance indicators, the way they are measured, target values, and financial penalties for the case of nonfulfillment [Sturm and Morris 2000]. However, in many cases—notably in private cloud settings, where the provider and the customers belong to the same organization—SLAs can be less formal and less detailed. It is also possible that there is no written SLA at all. But even in such a case, customers do have expectations about service quality, and SLOs may exist also without an SLA or even if the SLA is expressed in other terms [Serrano et al. 2015]. Failure to fulfill customer expectations damages the reputation of the CP, which in the long run will lead to customer churn and thus to profit loss. In this respect, SLA management is also closely related to trust and long-term partnership [Hani et al. 2015].

Hence, in all cases, it is the CP's financial interest to pay attention to the explicit or implicit SLAs and try to avoid or at least minimize the number of SLA violations [Beloglazov and Buyya 2012]. This constrains the consolidation opportunities because VM consolidation that is too aggressive and overbooking of PMs would degrade performance [Tesfatsion et al. 2014] and thus increase the probability of SLA violations [Tomás and Tordsson 2014]. However, measuring the underlying performance attributes and determining the fulfillment of the SLOs is a nontrivial task on its own [Garg et al. 2013].

We may differentiate hard and soft SLOs. A *hard* SLO must be fulfilled in any case. A *soft* SLO should be fulfilled as much as possible but may be violated (usually at the price of a financial penalty). From a problem formalization point of view, hard SLOs must be modeled as constraints, whereas soft SLOs are no constraints but the number of violations of a soft SLO must be minimized, and hence it will be part of the objective function.

Another distinction concerns the level of abstraction of the SLOs. Basically, we can differentiate between *user-level* SLOs describing quality metrics as observed by users (e.g., application response time, application throughput) and *system-level* SLOs defining the underlying technical objectives (e.g., system availability). Generally, user-level SLOs are more appropriate indicators of service performance; nevertheless, from a provider point of view, it is easier to control the system-level metrics, which will then indirectly determine the user-level metrics. For this reason, translating user-level objectives to system-level requirements is an important problem on its own [Chen et al. 2008].

An *SLA violation* occurs if one or more of the SLOs are not met. In many cases, this is the result of a situation in which a VM is not being allocated the required capacity (e.g., because of consolidation that is too aggressive). However, other factors, such as inappropriate sizing of VMs or inadequate elasticity solutions, can lead to an inability to serve requests within the boundaries stated in the SLA [Xiao et al. 2014].

3.7. Live Migration

Live migration of a VM from one PM to another makes it possible to react to the changing resource requirements of the VMs [Beloglazov and Buyya 2012]. For example, in times of low demand, several VMs can be consolidated to one PM so that other PMs can be switched off, thus saving energy. When the resource demand of the VMs increases, they can be migrated to other PMs with a lower load, thus avoiding SLA violations. For these reasons, VM migration is a key ingredient of dynamic VM placement schemes [Svärd et al. 2014].

On the other hand, VM migrations take time, create overhead, and can have adverse impact on SLA fulfillment [Rodero et al. 2012]. A VM migration may increase the load of both the source and the target PM, puts additional burden on the network, and makes the migrated VM less responsive during migration [Jung et al. 2010]. Therefore, it is important to keep the number of live migrations at a reasonable level.

Understanding the exact impact of live migration is a difficult problem on its own. A possible model for predicting the duration and overhead of live migration was presented by Verma et al. [2010, 2011]. According to their findings, migration increases the load of the source PM but not the load of the target PM. In contrast, other researchers also measured increased load on the target PM [Rodero et al. 2012]. The quest for a universally usable model of migration overhead is ongoing [Strunk 2012].

3.8. Actions of the CP

The CP has to update the VMs' placement in several cases:

—To react to a customer request [Shi et al. 2013]
—To react to critical situations [Gmach et al. 2009] and changes in system load [Sedaghat et al. 2013]
—In the course of a regular evaluation of the current placement to improve overall optimization objectives (see Section 3.9).

The first case is quite obvious. If a customer requests a new VM, it must be allocated on a PM or eCP. If a customer requests the cancellation of an existing VM, it must be removed from the hosting PM or eCP. Although rarely considered in the literature, a customer may also request a change in the parameters of a VM (e.g., resizing). In all of these cases, the CP must make a change to the current placement of VMs. This may also be a good occasion to review and reoptimize the placement of other VMs. For example, if a VM was removed upon the request of the customer, and the affected PM hosts only one more VM with a small load, then it may make sense to migrate that VM to another PM so that this PM can be switched off.

Often, a customer request consists of multiple VMs, such as VMs hosting the respective tiers of a multitier application [Iqbal et al. 2010]. Another important example is the case of elastic services: here, the number of VMs that take part in implementing the service changes automatically based on system load (autoscaling) [Li et al. 2012; Kecskemeti et al. 2011]. In such cases, it is important to consider the placement of the affected VMs jointly to avoid excessive communication costs [Alicherry and Lakshman 2012].

The CP must also react to unplanned situations, such as overloading of servers that may threaten SLA adherence [Wood et al. 2009], thermal anomalies [Rodero et al. 2012; Al-Qawasmeh et al. 2015], or breakdown of servers. Server unavailability may also be a planned situation (e.g., maintenance).

Besides the preceding reactive actions, a CP will also have to regularly review and potentially reoptimize the whole VM placement to find a better fit to the changed demand of the existing VMs, modified eCP rental fees, modified electricity prices, or other changes that did not require immediate action but made the placement suboptimal [Sedaghat et al. 2013; Svärd et al. 2014]. Such a review may be carried out at regular times (e.g., every 10 minutes), or it may be triggered by specific events. For instance, a CP can continuously monitor the load of its servers or the performance of the VMs, and whenever some load or performance indicator goes below or above specified thresholds, this may be a reason to reconsider the VM placement.

Reoptimizing the VM placement may consist of one or more of the following actions:

—Migration of a VM from one host to another
—Switching the state of a PM
—Starting/ending the rental of a VM from an eCP
—VM resizing.

Increasing or decreasing the resource allotment of a VM (“VM resizing”) can take multiple forms. In the case of VMs mapped on a PM owned by the CP, the virtual machine monitor (VMM) can be instructed to set the resources allocated to the respective VMs as necessary [Wood et al. 2009; Guazzone et al. 2012a; Verma et al. 2010]. In the case of VMs rented from eCPs, it may make sense to repack the application into VMs of different size (e.g., into a smaller number of larger VMs). This gives rise to an interesting balance between horizontal elasticity (number of VMs for the given service) and vertical elasticity (size of the VMs) [Sedaghat et al. 2013].

3.9. Objectives

VM placement is inherently a multiobjective problem [Gao et al. 2013; Tsamoura et al. 2013; Xu and Fortes 2010]. The following is a list of typical objectives for the CP:

—Monetary objectives:
  —Minimize fees paid to eCPs
  —Minimize operations costs
  —Amortize capital expenditures
  —Maximize income from customers
  —Avoid penalties
—Performance-related objectives:
  —Satisfy SLOs (availability, response time, makespan, etc.)
  —Minimize number of SLA violations
—Energy-related objectives:
  —Minimize overall energy consumption
  —Minimize number of active PMs
  —Minimize carbon footprint
—Technical objectives:
  —Minimize number of migrations
  —Maximize utilization of resources
  —Balance load among PMs
  —Minimize network traffic
  —Avoid overheating of hardware units.

Of course, not all of these goals are independent of each other—for example, several other objectives can be transformed to a monetary objective. Nevertheless, there are several independent or even conflicting objectives that VM placement should try to optimize. Given k objectives, to come to a well-defined optimization problem, one common technique is to constrain $k-1$ of the objectives and optimize the last one; another possibility is to optimize the weighted sum of the k objectives.

4. PROBLEM MODELS IN THE LITERATURE

A huge number of papers have been published about different versions of the VM allocation problem. In the following, we first give a categorization in Section 4.1, and then we review the problem models of the most important works. Most existing works concentrate on either the single-DC or multi-IaaS problem (defined in Section 4.1), which are quite different in nature; these problem formulations are discussed in Sections 4.2 and 4.3, respectively. Finally, some other problem models are described in Section 4.4.

4.1. Important Special Cases and Subproblems

The problem described in Section 3 is very general. Most authors investigated special cases or subproblems, the most popular of which are presented next. It should be noted that these problem variants are not necessarily mutually exclusive—a given work may deal with a combination of them.

4.1.1. The Single-DC Problem. The subproblem that has received the most attention is the single-DC problem. In this case, the CP has a single DC with a number of PMs, and there are no eCPs. Usually, the number of PMs is assumed to be high enough to serve all customer requests. Typical objectives are optimizing the utilization of resources and minimizing overall energy consumption, subject to performance constraints (SLAs). Since all PMs are in the same DC, network bandwidth is often assumed to be uniform and sufficiently high so that it can be ignored.

4.1.2. The Multi-IaaS Problem. In this case, the CP does not own any PMs; it uses only leased VMs from multiple IaaS providers. Since there are no PMs, all concerns related to them—states and state transitions, sharing of resources among multiple VMs, load-dependent power consumption—are void. Power consumption plays no role; the main

goals are minimizing the monetary costs associated with VM rental and maximizing performance. Since data transfer between the different IaaS providers can become a bottleneck, this also has to be taken into account.

It is important to mention that the literature on the multi-IaaS problem is mostly unrelated to the literature on the single-DC problem. On the one hand, this is natural because the two problems are quite different. On the other hand, a hybrid CP must solve a combination of these two problems. This is why we include both of them in our article, and we expect increased convergence between them in the future.

4.1.3. The One-Dimensional VM Placement Problem. In this often-investigated special case, only the computational demands and computational capacities are considered, and no other resources. Moreover, the CPU is taken to be single core, making the problem truly one-dimensional.

The question whether one or more dimensions are taken into account is independent of whether own PMs or eCPs are used. In other words, the one-dimensional VM placement problem can be a special case of the single-DC, the multi-IaaS, or other problem formulations.

4.1.4. The On/Off Problem. In this case, each PM has only two states: On and Off. Furthermore, the power consumption of PMs that are Off is assumed to be 0, whereas the power consumed by PMs that are On is the same positive constant for each PM, and dynamic power consumption is not considered. The transition between the two power states is assumed to be instantaneous. As a consequence, the aim is simply to minimize the number of PMs that are On. This is an often-investigated special case of the single-DC problem.

4.1.5. Online Versus Offline Optimization. As mentioned in Section 3.8, the CP must react immediately to customer requests. This requires local modifications: allocating a new VM to a host, possibly turning on a new host if necessary, or deallocating a VM from a host, possibly switching the host to a low-energy state if it becomes empty. Finding the best reaction to the customer request in the given situation is an *online* optimization task.

On the other hand, the CP can also (e.g., on regular occasions) review the status of all VMs and hosts, and possibly make global modifications (e.g., migrating VMs between hosts). Finding the best new configuration is an *offline* optimization task.

These are two distinct tasks for which a CP may use two different algorithms.

It should be noted that there is some ambiguity in the literature on the terminology used to differentiate between the preceding two cases, and the terms *online* and *offline* are used by some authors to describe other problem characteristics. We use these notions in this sense because this is in line with their generally accepted meaning in the theory of algorithms.

4.1.6. Placement Tasks. Closely related to online versus offline optimization is what we may call the *placement task*. On the one hand, (1) initial placement and (2) placement reoptimization must be differentiated: the former determines a placement for a new set of VMs, whereas the latter optimizes an existing placement. (The key difference is that placement reoptimization must use migrations, which is not necessary for initial placement.) On the other hand, based on the set of VMs for which the placement is determined, the following three different levels can be distinguished: (1) all VMs of the CP, (2) a set of coupled VMs (e.g., the VMs implementing a given service), or (3) a single VM. Since these are two independent dimensions, we get six possible placement tasks; all of them are meaningful, although some are rather rare (e.g., initial placement of all VMs occurs only when a new DC starts its operation). It should also be noted that

some works addressed multiple placement tasks, such as initial placement of a single VM and placement reoptimization of all VMs.

4.1.7. The Load Prediction Problem. When the CP makes some change in the mapping of VMs or the states of PMs at time instance t_0, it can base its decision only on its observations of VM behavior for the period $t \leq t_0$; however, the decision will have an effect only for $t > t_0$. The CP could make ideal decisions only if it knew the future resource utilization of the VMs. Since these are not known, it is an important subproblem to *predict* the resource utilization values of the VMs or their probability distributions, at least for the near future [Xiao et al. 2013].

Load prediction is seen by some authors as an integral part of the VM placement problem, whereas others do not consider it, either because VM behavior is assumed to be constant (at least in the short run) or it is assumed that load prediction is done by a separate algorithm. Load prediction may or may not be considered, independently from the types of resources (i.e., also within the single-DC or multi-IaaS problem).

4.2. The Single-DC Problem

The single-DC problem also received a lot of attention before the cloud computing age, with the main objective of achieving good utilization of physical resources in a DC. Early works include Muse, a resource management system for hosting centers [Chase et al. 2001], approaches to using dynamic voltage scaling for power management of server farms [Horvath et al. 2007] and to dynamic provisioning of multitier Internet applications [Urgaonkar et al. 2005], and first results on consolidation using VM migration [Khanna et al. 2006]. The term *load unbalancing* was coined to describe the objective of consolidating load on a few highly utilized PMs instead of distributing them among many PMs with low utilization [Pinheiro et al. 2001]. Since about 2007, as virtualized DCs have become ever more prevalent, the amount of research on resource management in DCs has seen significant growth [Bobroff et al. 2007; Batista et al. 2007; Verma et al. 2008a]. These works already exhibited all of the important characteristics of the problem: consolidation of the VMs on fewer PMs using migrations, taking into account service levels and load fluctuations.

In recent years, the handling of SLA violations has become more sophisticated and energy consumption has become one of the most crucial optimization objectives. For example, the work of Beloglazov and Buyya [2010a, 2012], Beloglazov et al. [2012], and Guazzone et al. [2012a, 2012b] has focused specifically on minimizing energy consumption.

Energy minimization can be primarily achieved by minimizing the number of active servers; thus, it is no wonder that many works focused only on this and ignored the dynamic power consumption of PMs (leading to the special case of the On/Off problem). Exceptions include the work of Jung et al. [2010], which treated dynamic power consumption as a linear function of CPU load, as well as the nonlinear function used by Guazzone et al. [2012a] and the table-based approach used in pMapper [Verma et al. 2008a].

Most works on the single-DC problem consider only the CPU capacity of the PMs and the computational demand of the VMs, but no other resources, reducing the problem to a single dimension. Several authors mentioned this deficiency as an area for future research [Beloglazov and Buyya 2010b; Guazzone et al. 2012a]. Only few works also take into account memory [Ribas et al. 2012; Shi et al. 2013] or memory and I/O as further dimensions [Mishra and Sahoo 2011; Tomás and Tordsson 2014; Wood et al. 2009]. Moreover, the sharing of cores of multicore CPUs was hardly addressed explicitly. For example, Beloglazov and Buyya [2012] model a multicore CPU by means of a single-core CPU with capacity equal to the sum of the capacities of the cores of the original

multicore CPU. Another extreme is the approach of Ribas et al. [2012], which does consider multicore CPUs, but only the number of cores is taken into account—their capacity is not.

The majority of these works did not address the load prediction problem. A notable exception is the early work of Bobroff et al. [2007], which uses a stochastic model to probabilistically predict the future load of a VM based on past observations. More recently, Guenter et al. [2011] used linear regression for similar purposes in a slightly different setting without virtualization. Beloglazov and Buyya [2013] introduce a Markov chain approach for a related but perhaps somewhat simpler problem: to detect when a PM becomes overloaded.

Concerning the investigated SLAs, most works consider the number of occasions when a server is overloaded [Beloglazov and Buyya 2010a; Beloglazov et al. 2012; Bobroff et al. 2007], which indirectly lead to SLA violations. Only a few works directly considered the response time [Guazzone et al. 2012a] or waiting time [Salehi et al. 2012] as specific metrics with quantitative QoS requirements.

The main characteristics of some representative works are summarized in Table I. The meaning of the table's columns is explained next. A full circle means that the cited work explicitly deals with the given characteristic as part of their problem formulation and algorithms; an empty circle means that the given work does not explicitly address it:

—*Resources*: The types of resources of VMs and PMs that are taken into account
  —*CPU*: Computational capacity of the PMs and computational load of the VMs are taken into account.
  —*Cores*: Individual cores of a multicore processor are differentiated.
  —*Other*: At least one resource other than the CPU (e.g., memory) is also taken into account.
—*Energy*: The way energy optimization is supported by the given approach
  —*Switch off*: The given approach aims at emptying PMs so that they can be switched to a low-power state.
  —*Dynamic power*: The dynamic power consumption of the PMs also is taken into account.
—*Placement*: The kind of placement task addressed by the given work
  —*Initial*: The initial placement of the VMs is determined.
  —*Reoptimization*: An existing placement is optimized.
  —*All VMs*: The placement of all VMs in the DC is determined.
  —*VM set*: The placement of a set of coupled VMs that together form a service is determined.
  —*One VM*: The placement of a single VM is determined.
—*SLA*: The way in which SLAs are handled (see Section 3.6)
  —*Soft*: Soft SLAs are supported.
  —*User level*: User-level SLAs are supported.
  —*Priorities*: VMs may have different priorities.
—*Other*: Some other important aspects
  —*Different PMs*: Differences in the capacity and/or power consumption of PMs is leveraged to find the best VM-to-PM mapping.
  —*Migration*: The approach leverages migration of VMs between PMs.
  —*Migration cost*: Migration costs are taken into account and must be minimized.
  —*Data transfer*: The communication between VMs is taken into account.
  —*Load prediction*: The future load of the VMs is predicted by the approach based on past observations.

Table I. Characteristics of Problem Models in the Single-DC Problem

	Resources			Energy		Placement					SLA			Other				
Paper	CPU	Cores	Other	Switch off	Dynam. power	Initial	Reoptimization	All VMs	VM set	One VM	Soft	User-level	Priorities	Different PMs	Migration	Migration cost	Data transfer	Load predict.
Batista et al. [2007]	●	○	○	○	○	●	○	○	●	○	○	●	○	●	○	○	●	○
Beloglazov et al. [2012]	●	○	○	●	●	●	●	●	○	●	●	○	○	●	●	○	○	○
Beloglazov and Buyya [2012]	●	○	○	●	●	●	●	●	○	●	●	○	○	●	●	●	○	○
Beloglazov and Buyya [2013]	●	○	○	○	○	○	○	○	○	○	●	○	○	○	●	○	○	●
Biran et al. [2012]	●	○	●	○	○	●	○	●	○	○	○	○	○	●	○	○	●	○
Bobroff et al. [2007]	●	○	○	●	○	○	●	●	○	○	●	○	○	○	●	○	○	●
Breitgand and Epstein [2011]	●	○	○	○	○	●	●	○	●	○	○	○	●	●	●	●	○	○
Das et al. [2008]	○	○	○	●	○	○	●	●	○	○	●	●	○	○	○	○	○	○
Guazzone et al. [2012a]	●	○	○	●	●	○	●	●	○	○	●	●	○	●	●	○	○	○
Guenter et al. [2011]	○	○	○	●	○	○	○	○	○	○	●	○	○	○	○	○	○	●
He et al. [2012]	○	●	●	○	○	○	●	●	○	○	○	○	○	○	●	○	○	○
Jayasinghe et al. [2011]	●	○	●	○	○	●	○	○	●	○	○	○	○	●	○	○	●	○
Jung et al. [2010]	●	○	○	●	●	○	●	●	○	○	●	●	○	○	●	●	○	●
Li et al. [2011b]	○	○	○	○	○	●	○	○	●	○	○	○	○	●	○	○	○	○
Mishra and Sahoo [2011]	●	○	●	●	○	●	●	●	○	○	○	○	○	○	●	○	○	○
Ribas et al. [2012]	○	●	●	●	○	●	○	●	○	○	○	○	○	●	○	○	○	○
Salehi et al. [2012]	○	○	○	●	○	○	●	●	○	●	●	●	●	○	●	○	○	○
Shi et al. [2013]	●	○	●	●	○	○	●	●	○	●	○	○	○	●	●	●	○	○
Song et al. [2014]	●	○	●	●	○	○	●	●	○	○	●	○	○	○	●	●	○	●
Srikantaiah et al. [2009]	●	○	●	●	●	●	●	●	○	●	○	●	○	○	●	○	○	○
Tomas and Tordsson [2014]	●	○	●	○	○	●	○	○	●	●	●	○	○	●	○	○	○	●
Verma et al. [2008a]	●	○	○	●	●	○	●	●	○	○	○	●	○	●	●	●	○	○
Verma et al. [2009]	●	○	○	●	○	○	●	●	○	○	●	○	○	●	●	○	○	●
Wood et al. [2009]	●	○	●	○	○	○	●	●	○	○	○	○	○	●	●	●	○	●
Xiao et al. [2013]	●	○	●	●	○	○	●	●	○	○	●	○	○	○	●	●	○	●

As can be seen in Table I, there are many differences between the approaches presented in the literature. In fact, it is hard to find two that address exactly the same problem. Of course, some basic properties are typical of most approaches (e.g., the CPU is considered in almost all works, as well as the possibility to migrate VMs and to switch off unused PMs). Other characteristics such as the sharing of individual cores of a multicore CPU among VMs or communication between VMs are still largely unexplored.

Of course, Table I should not be seen as a valuation of these works (assuming that more filled circles indicate a higher "score"). Approaches that tackle a limited version of the problem can be highly valuable if that problem is practically meaningful and the approach addresses it in an effective and efficient way. It is also important to mention that we focus here only on problem models and algorithms, but some works include many other aspects. Indeed, some works describe complete systems that are

successfully applied in practice, such as Mistral [Jung et al. 2010], Muse [Chase et al. 2001], pMapper [Verma et al. 2008a], and Sandpiper [Wood et al. 2009].

4.3. The Multi-IaaS Problem

As already mentioned, the multi-IaaS problem is quite different from the single-DC problem. In the multi-IaaS problem, the utilization and state of PMs, as well as their energy consumption, are not relevant. On the other hand, monetary costs related to the leasing of VMs from eCPs appear as a new factor to consider. In fact, some works consider quite sophisticated leasing fee structures—for example, VMs reserved for longer periods may be cheaper than on-demand VMs [Genez et al. 2012], or the costs may consist of a fixed rental fee and usage-based variable fees for the used resources [Lampe et al. 2012].

In many formulations of the multi-IaaS problem, the entities that need to be mapped to resources are not VMs but (computational) tasks. However, this is not really a significant conceptual difference: also in the single-DC problem, the actual goal is to map applications or components of applications to resources, and VMs are just wrappers that facilitate the safe co-location of applications or components of applications on the same resources and their migration.

More importantly, communication and dependencies among the tasks are often considered important ingredients of the multi-IaaS problem [Bittencourt et al. 2012b; Genez et al. 2012; Oliveira et al. 2012], in contrast to the single-DC problem where communication among VMs is hardly considered.

In the multi-IaaS problem, the tasks and their dependencies are often given in the form of a directed acyclic graph (DAG), in which the vertices represent the tasks and the edges represent data transfer and dependencies at the same time [Calheiros and Buyya 2014]. Scientific workflows are popular examples of complex applications that are well suited for a DAG representation [Oliveira et al. 2012; Wu et al. 2010]. The resulting problem, often called the *workflow scheduling problem* [Bardsiri and Hashemi 2012], has the advantage of solid mathematical formalism using graph theory; moreover, it is similar to other multiresource scheduling problems (e.g., multiprocessor scheduling) so that a rich arsenal of available scheduling techniques can be applied to it [Pinedo 2008]. Besides minimizing cost, the other objective of such scheduling problems is to minimize the makespan of the workflow—that is, the time it takes from the start of the first task to the finish of the last task.

The main characteristics of some representative works are summarized in Table II. The meaning of full versus empty circles is the same as in Table I. The meaning of the table's columns, where different from those of Table I, is explained next:

—*Scheduling*: The way scheduling-related aspects are modeled
  —*Dependencies*: Dependencies between tasks arising from data transfer are considered.
  —*Makespan*: Minimization of the workflow's makespan is either an explicit objective or there is an upper bound on the makespan.
—*Costs*: The kinds of monetary costs of leased VMs that the approach takes into account
  —*Long-term rental*: These are discounted fees for VMs that are rented for a long term (e.g., multiple months).
  —*On demand*: These fees are either proportional to the time the VM is used or charged for small time quanta (e.g., hourly), based on the number of time quanta the VM is used.
  —*Usage based*: These fees are proportional to the used amount of some resource, such as the number of transferred bytes to/from a VM.
—*Other*: Some miscellaneous aspects

Table II. Characteristics of Problem Models in the Multi-IaaS Problem

	Resources			Scheduling		Costs			Other	
Paper	CPU	Cores	Other	Dependencies	Makespan	Long-term rental	On-demand	Usage-based	Migration	Load prediction
Bittencourt et al. [2012b]	○	○	○	●	●	○	●	○	○	○
Candeia et al. [2010]	●	○	○	○	●	○	●	○	●	○
Genez et al. [2012]	○	●	○	●	●	●	●	○	○	○
Lampe et al. [2012]	○	○	●	○	●	●	○	●	○	○
Li et al. [2011a]	●	○	○	○	○	○	●	○	●	○
Oliveira et al. [2012]	●	○	●	●	●	○	●	●	○	●
Pandey et al. [2010]	●	○	○	●	○	○	●	●	○	○
Tordsson et al. [2012]	○	○	○	○	○	○	●	○	○	○
Tsamoura et al. [2013]	○	○	○	○	●	○	●	○	○	○
Villegas et al. [2012]	●	○	○	○	●	○	●	○	○	●

—*Migration*: The approach leverages migration of tasks between VMs or between eCPs.
—*Load prediction*: The future load of the tasks is predicted by the approach based on past observations.

As can be seen from Table II, computational capacity and computational load, which are mostly considered one dimensional (i.e., without accurate modeling of multicore CPUs) are also the focus of most works in the multi-IaaS context, just like in the case of the single-DC problem. Makespan minimization and the minimization of on-demand rental costs are considered in most works. The other aspects are rarely handled. Again, it is interesting to note how the used problem formulations are different.

4.4. Other Problem Formulations

Although most of the relevant works fall into either the single-DC or multi-IaaS category, there are a few works that address some other, more general problems.

4.4.1. Multi-DC. An important generalization of the single-DC problem is the *multi-DC* problem, in which the CP possesses multiple DCs. For an incoming VM request, the CP must first decide in which DC the new VM should be provisioned and then on which PM of the selected DC. Although the second step is the same as the single-DC problem, choosing the most appropriate DC may involve completely different decision making [Alicherry and Lakshman 2012]. A possibility is to consider the different power efficiency and carbon footprint of the different DCs, taking into account that different DCs may have access to different energy sources—for example, some DCs may be able to better leverage renewing energy sources. In an attempt to optimize the overall carbon footprint, the CP may prefer to utilize such "green" DCs as much as possible [Khosravi et al. 2013].

4.4.2. Hybrid Cloud. In most works that address hybrid cloud setups, the CP owns one DC and also has some eCPs at its disposal. This can be seen as a common generalization of the single-DC and multi-IaaS problems.

Casalicchio et al. [2013] address this problem with an emphasis on the single-DC subproblem. The PMs are explicitly modeled, migrations between PMs are allowed

but incur a cost, and there is a sophisticated handling of SLAs, but communication and dependencies among VMs are not handled, similarly to many formulations of the single-DC problem.

In contrast, the approach of Bittencourt et al. shows more similarity to formulations of the multi-IaaS problem. Here, dependencies among the tasks are given in the form of a DAG, there is a hard deadline on the makespan, and the objective is to minimize the total VM leasing costs, as is common in workflow scheduling. The own DC of the CP is modeled as a special eCP, offering free resources, but only in limited quantity [Bittencourt and Madeira 2011; Bittencourt et al. 2012a].

Bossche et al. [2010] use a similar approach, which is largely based on the multi-IaaS problem, and own DCs are modeled as special eCPs offering free resources in limited quantity. They explicitly allow having more than one own DC, so this can be seen as a common generalization of the multi-DC and multi-IaaS problems. On the other hand, the model uses a number of restrictions—for example, communication and dependencies between VMs are not supported, and neither are migration of VMs or aspects related to power consumption.

5. OVERVIEW OF PROPOSED ALGORITHMS

From a theoretical point of view, we must differentiate between exact algorithms that are guaranteed to always deliver the optimum and heuristics that do not offer such a guarantee. Although the majority of the proposed algorithms are heuristics, some exact algorithms have been proposed as well, so it makes sense to review the two groups separately.

As mentioned previously, most of the literature deals with either the single-DC or multi-IaaS problem, and these two are quite different. Interestingly, the exact methods proposed for the two problems are very similar, and hence we review them together. On the other hand, the heuristics proposed for the two problems are quite different, so we review them separately.

5.1. Exact Algorithms

In most cases, the exact algorithm consists of formulating the problem in terms of some mathematical programming formalism and using an existing solver to solve the mathematical program.

By far, integer linear programming (ILP) seems to be the most popular way to express both the single-DC [Batista et al. 2007; Guenter et al. 2011; Li et al. 2011b] and multi-IaaS problems [Genez et al. 2012; Lampe et al. 2012; Li et al. 2011a] or even their common generalization [Bossche et al. 2010] as a mathematical program. Several authors found that even the special case of ILP in which each variable is binary (binary integer programming (BIP)) is sufficient to express the constraints of the problem in a natural way [Bossche et al. 2010; Lampe et al. 2012; Li et al. 2011a, 2011b].

Some authors preferred to use nonlinear constraints, leading to a mixed-integer nonlinear programming (MINLP) formulation [Guazzone et al. 2012b; Konstanteli et al. 2014] or a pseudo-Boolean (PB) formulation with binary variables and a combination of linear and nonlinear constraints [Ribas et al. 2012].

For all of these mathematical programs, appropriate solvers are available, both as commercial and as open-source software packages. In each case, the solver will deliver optimal results, but its worst-case runtime is exponential with respect to the size of the input, so solving large-scale problem instances takes much too long. Most researchers turned to heuristics for this reason.

It is important to mention that an ILP formulation can be useful in devising a heuristic. Removing the integrality constraint, the resulting linear programming (LP) formulation can be solved in polynomial time. The result obtained this way may not be

an integer, but in some cases a rounding method can be used to turn it into an integer solution with near-optimal cost [Batista et al. 2007; Genez et al. 2012].

5.2. Heuristics for the Single-DC Problem

Several authors observed the similarity between the VM placement problem in a single DC and the well-known bin-packing problem, in which objects of given weight must be packed into a minimum number of unit-capacity bins. Indeed, if only one dimension, such as the computational demand of the VMs and the computational capacity of the PMs, is considered, and the aim is to minimize the number of PMs that are turned on, the resulting problem is very similar to bin packing. There are some simple but effective heuristics for bin packing, such as First Fit (FF), in which each object is placed into the first bin where it fits, Best Fit (BF), in which each object is placed in the bin where it fits and the remaining spare capacity is minimal, and Worst Fit (WF), in which each object is placed in the bin where it fits and the remaining spare capacity is maximal. Despite their simplicity, these algorithms are guaranteed to deliver results that are at most 70% off the optimum [Dósa and Sgall 2013, 2014]. This approximation ratio can be improved if the objects are first sorted in decreasing order of their weights, leading to modified algorithms such as First Fit Decreasing (FFD) and Best Fit Decreasing (BFD). Specifically, if OPT denotes the optimal number of bins, then FFD is guaranteed to use no more than $11/9\ OPT + 6/9$ bins [Dósa 2007].

These simple bin-packing heuristics can be easily adapted to the VM placement problem. Indeed, the use of FF has been suggested [Bobroff et al. 2007], just like BF [Beloglazov and Buyya 2012], WF [Jung et al. 2010; Li et al. 2011b; Tomás and Tordsson 2014], FFD [Verma et al. 2008a, 2009], and BFD [Beloglazov and Buyya 2010a; Beloglazov et al. 2012; Guazzone et al. 2012a]. However, it should be noted that the approximation results concerning these algorithms on bin packing do not automatically carry over to the more complicated VM placement problem [Mann 2015a]. The application of semionline and relaxed online bin-packing algorithms has also been proposed [Song et al. 2014; Xiao et al. 2014].

Metaheuristics have also been suggested, such as simulated annealing [Hyser et al. 2008], genetic algorithms [Gmach et al. 2008], and ant colony optimization [Gao et al. 2013].

Some authors proposed proprietary heuristics. Some of them are simple greedy algorithms [Salehi et al. 2012; Wood et al. 2009] or straightforward selection policies [Batista et al. 2007; Beloglazov and Buyya 2010a; Beloglazov et al. 2012; Shi et al. 2013]. Others are rather complex. For example, the algorithm of Jung et al. [2010] first determines a target mapping by means of a WF-like heuristic but then uses an A^* tree traversal algorithm to create a reconfiguration plan, taking into account not only the adaptation costs but also the cost of running the algorithm itself (which means that search space exploration is restricted if the algorithm has already run for a long time); moreover, this algorithm is carried out in a hierarchical manner on multiple levels. Mishra and Sahoo [2011] categorize both PMs and VMs according to what kind of resource is used by them most (from the three investigated dimensions, which are CPU, memory, and I/O) into so-called resource triangles, and attempt to match them on the basis of complementary resource triangles (e.g., a VM that uses the CPU most should be mapped on a PM where the CPU is the least-used resource), at the same time also taking into account the utilization levels. Verma et al. [2009] devised an algorithm that starts by analyzing the workload time series of the applications to determine an envelope of the time series that captures the bulk and the peak of the distribution. This information is then used to cluster the applications on the basis of correlating peaks, and then the application clusters are spread evenly on the necessary number

of PMs. However, none of these sophisticated heuristics offers performance guarantees in terms of approximation factors.

A different approach is to regard the VM placement problem as a *control* task, in which a controller tries to balance the utilization of the PMs between the conflicting objectives of minimizing power consumption and keeping performance levels, and to apply control-theoretic methods. This includes fuzzy control techniques [Salehi et al. 2012] and distributed PID controllers [Tomás and Tordsson 2014].

5.3. Heuristics for the Multi-IaaS Problem

The heuristic algorithms that have been proposed for the multi-IaaS problem are quite heterogeneous. The simplest algorithms include list scheduling [Bittencourt et al. 2012b], greedy provisioning and allocation policies [Villegas et al. 2012], greedy scheduling and clustering algorithms [Oliveira et al. 2012], and simple proprietary heuristics [Candeia et al. 2010]. Metaheuristics have also been suggested, such as particle swarm optimization [Pandey et al. 2010]. Additionally, more sophisticated algorithms have been proposed, such as those based on existing algorithms for the knapsack problem [Lampe et al. 2012].

The preceding algorithms, whether simple or sophisticated, offer no performance guarantees, or at least none has been proven. An exception is the work of Tsamoura et al. [2013] addressing a multiobjective optimization problem, in which makespan and cost are minimized simultaneously. The aim is to find Pareto-optimal solutions in the time–cost space, and the bids of eCPs are in the form of time–cost functions. Drawing on earlier results [Papadimitriou and Yannakakis 2001], an approximation algorithm with pseudopolynomial runtime can be devised.

5.4. Algorithms for Other Problem Formulations

As mentioned in Section 4.4, there are few works considering other problem formulations, such as the multi-DC problem or hybrid cloud setups, and these works are similar to either the single-DC or multi-IaaS problem. Accordingly, the algorithms that have been proposed for these problem formulations are similar to the ones for the other problem variants.

In particular, Binary Integer Programming has been suggested to optimally solve the task allocation problem in a hybrid cloud scenario [Bossche et al. 2010]. Hill climbing has also been used as a simple heuristic [Casalicchio et al. 2013], as well as proprietary heuristics [Bittencourt and Madeira 2011]. Heuristics inspired by bin packing play a role here as well, such as FF [Khosravi et al. 2013], and Mills et al. [2011] compare several bin-packing–style heuristics in a multi-DC setup.

5.5. Evaluation of Algorithms

Most papers also provide some evaluation of the algorithms they propose. In most cases, this evaluation is done empirically, but there also are some examples of rigorous mathematical analysis.

5.5.1. Rigorous Analysis. Tsamoura et al. [2013] proved the correctness and complexity of their algorithms: an exact polynomial-time algorithm for a special case and an approximation algorithm with pseudopolynomial runtime for the general case, albeit for a rather uncommon problem formulation.

For some restricted problem versions, polynomial-time approximation algorithms have been presented with rigorously proven approximation guarantees [Alicherry and Lakshman 2012, 2013; Breitgand and Epstein 2012; Mann 2015c; Song et al. 2014].

Guenter et al. [2011] proved an important property of the linear program that they proposed: that its optimal solution will be integral, without explicit integrality constraints, thus allowing the use of an LP solver instead of a much slower ILP solver.

5.5.2. Empirical Evaluation. In many cases, the evaluation was carried out using simulation. There are simulators specifically for cloud research, such as CloudSim [Calheiros et al. 2011], but many researchers used their own simulation environments. Relatively few researchers tested their algorithms in a real environment [Das et al. 2008; Jung et al. 2010; Koller et al. 2010; Liu et al. 2009; Meng et al. 2010; Nathuji and Schwan 2007; Verma et al. 2009, 2011; Wood et al. 2009] or using a combination of real hardware and simulation [Rodero et al. 2012; Svärd et al. 2014; Tomás and Tordsson 2014; Villegas et al. 2012; Zhu et al. 2009]. However, it must be noted that in most of these cases, the "real" environment used for evaluation was rather small (e.g., just a handful of PMs and VMs). Apparently, most researchers do not have the possibility to perform experiments on large-scale real systems.

As a compromise between pure simulation and a real evaluation environment, several researchers used traces from real applications and real servers. Some research groups of industry players used traces from their own infrastructure [Gmach et al. 2008; Guenter et al. 2011; Verma et al. 2008a; Zhu et al. 2009]. Others used publicly available workload traces, such as those from the Parallel Workloads Archive (http://www.cs.huji.ac.il/labs/parallel/workload/) of the Hebrew University [Feitelson et al. 2014; Jiang et al. 2012; Salehi et al. 2012], the Grid Observatory (http://grid-observatory.org/) [Rodero et al. 2012], PlanetLab (http://www.planet-lab.org/) [Beloglazov and Buyya 2012, 2013], or workload traces made available by Google (https://code.google.com/p/googleclusterdata/) [Reiss et al. 2012; Ribas et al. 2012]. A related approach, taken by several researchers, was to use a Web application with real Web traces. For example, RUBiS, a Web application for online auctions [Cecchet et al. 2003], has been used by multiple researchers with various Web server traces [Jung et al. 2010; Tomás and Tordsson 2014]; Wikipedia traces were also used [Ferrer et al. 2012]. Other benchmark applications used include the NAS Parallel Benchmarks (http://www.nas.nasa.gov/publications/npb.html) [Koller et al. 2011; Moreno-Vozmediano et al. 2011; Tordsson et al. 2012; Verma et al. 2011], the BLAS linear algebra package (http://www.netlib.org/blas/) [Verma et al. 2011], and the related Linpack benchmark (http://netlib.org/benchmark/hpl/) [Das et al. 2008; Verma et al. 2008a].

6. DETAILS OF SOME SELECTED WORKS

Because of the sheer volume, it is impossible to provide a detailed description of all works in the field. However, we selected some of the most influential and most interesting works and give more details about them in the following. "Most influential" has been determined based on the yearly average number of citations that the given paper has received according to Google Scholar (http://scholar.google.com), as of February 2015, and this list has been extended with some other works that are—in our opinion—also of high importance to the field.

6.1. One-Dimensional Dynamic VM Consolidation in a Single DC

According to the above metric, the most influential papers are those of Beloglazov and Buyya from the University of Melbourne (one of those papers is joint work with Abawajy). They address the single-DC problem, focusing on the single dimension of CPU capacity of PMs and CPU load of VMs. The main optimization objective is to consolidate the workload on the minimal number of PMs with the aim of minimizing

energy consumption. As a secondary objective, also the number of migrations should be kept low. The authors' early works focus on analyzing the context of and requirements towards such an optimization framework, as well as architectural considerations and preliminary results on some efficient optimization heuristics [Beloglazov and Buyya 2010a, 2010b].

Those heuristics are presented in more detail in a later paper [Beloglazov et al. 2012]. The main idea is to first remove all VMs from lightly used PMs so that they can be switched off and also remove some VMs from overloaded PMs so that they will not be overloaded. In a second phase, a new accommodating PM is searched for the removed VMs. The latter subproblem is seen as a special version of the bin-packing problem, in which the bins may have differing sizes (different PM capacities) and prices (different energy efficiency of the PMs). For this problem, the authors developed the Modified Best Fit Decreasing (MBFD) heuristic, which considers the VMs in decreasing order of load and allocates each of them to the PM with the best energy efficiency that has sufficient capacity to host it. For the problem of selecting some VMs to migrate off an overloaded PM, the authors consider several heuristics. The Minimization of Migrations (MM) policy selects the minimum number of VMs that must be removed to let the PM's load go back to the normal range. The Highest Potential Growth (HPG) policy selects the VMs that have the lowest ratio of current load to requested load. Finally, the Random Choice (RC) policy selects the VMs to be removed randomly. The authors used the CloudSim framework to simulate a DC with 100 PMs and 290 VMs to evaluate the presented heuristics and compare them to a Non-Power Aware (NPA) method, one using DVFS only, and a Single-Threshold (ST) VM selection algorithm. The simulation results, accompanied by a detailed statistical analysis, demonstrate the superiority of the presented methods with respect to energy consumption, number of SLA violations, and number of migrations. From the presented VM selection methods, the MM heuristic proved best.

In a related paper, the same authors provide a mathematical analysis of some rather restricted special cases or subproblems of the single-DC problem [Beloglazov and Buyya 2012]. In particular, they provide optimal offline and online algorithms for the problem of when to migrate a VM off from a PM and prove an upper bound for the competitive ratio of the optimal online algorithm for the case of n homogeneous PMs. Besides, they also consider some adaptive heuristics for dynamic VM consolidation. The problem is the same as the one considered in the other works of the authors, and the algorithms are also similar but are now adaptive: instead of using fixed thresholds for determining underutilization and overutilization, the thresholds now adapt to the variability of the VMs' load. For this, several methods are considered: Median Absolute Deviation (MAD), Interquartile Range (IQR), Local Regression (LR), and Robust Local Regression (RLR). The performance of the algorithms is evaluated again using CloudSim, but this time with a simulated DC with 800 heterogeneous PMs and real workload traces from PlanetLab. The authors also carried out a very thorough statistical analysis to come to the conclusion that the LR method outperforms the others in terms of energy consumption and SLA violations.

Finally, yet another paper of the same authors looks at one specific subproblem of VM consolidation: how to decide when a PM is overloaded and, consequently, when VMs should be removed [Beloglazov and Buyya 2013]. Two conflicting goals are taken into account: on the one hand, the time when the host is overloaded should be minimized to avoid performance degradation and SLA violation; on the other hand, the overload detection method should signal an overload only if absolutely necessary to keep the utilization high and avoid unnecessary migrations. The authors devise a method using Markov chains for stationary workloads, which can also be applied to nonstationary workloads by using the Multisize Sliding Window workload estimation

technique. Simulation results on PlanetLab traces demonstrate the good performance of the proposed method.

Bobroff et al. [2007] from IBM Research investigated a similar problem: they also aim at minimizing the number of active PMs and the number of SLA violations by carefully consolidating VMs to PMs in a single DC. The problem is one dimensional here as well, with the CPU being the single investigated resource, and SLA violations are assumed to happen if the CPU of a PM is overloaded. As a generic solution framework, the authors propose the Measure-Forecast-Remap cycle, in which the workload consumption of VMs is measured, based on which their future resource demand is forecast, and a new VM-to-PM mapping is generated. This cycle is iterated at regular time intervals of length τ. (In the practical examples, τ is 15 minutes.) The Remap phase is based on the similarity to the bin-packing problem and makes use of an FF heuristic. The strength of the paper lies in the solution for the Forecast phase (the load prediction problem, in our terminology). It is based on a sophisticated time series analysis, aiming to identify the principal periodic components of the load distribution based on past data. As a result, the future load can be estimated along with the distribution of the prediction error. This allows consolidation with a given upper limit on the allowed probability of server overload.

Another similar work is pMapper by Verma et al. [2008a], from IBM India and IIT Delhi. The aim here is to optimize the mapping of VMs to PMs with respect to energy consumption and number of migrations. Another similarity is the one-dimensional nature of the problem, considering only CPU capacity and CPU load. Besides trying to switch off PMs, the authors emphasize dynamic power consumption. Interestingly, they find that utilization does not determine power consumption and argue that this prohibits the use of global optimization techniques. Instead, local power efficiency characteristics are formulated that seem to hold in practice and can be exploited for local optimization techniques. Based on these insights, three algorithms are presented. The first one, min Power Parity (mPP), is a variation of the FFD heuristic: it considers VMs in decreasing order of CPU load and puts each VM into the PM with sufficient capacity that offers the best energy efficiency. The weakness of this method is that it can lead to a prohibitively large number of migrations. Hence, the second algorithm, min Power Placement with History (mPPH), enhances mPP by taking into account the starting allocation so that unnecessary migrations can be avoided. The third algorithm, pMaP, goes one step further in decreasing the number of migrations: it uses mPPH to generate a recommended new placement but actually performs only those migrations that improve the overall energy—migrations trade-off. The algorithms were implemented in the framework of the pMapper system and tested with a simulator using server utilization traces from a real DC. The authors' algorithms were compared to a non-power-aware load balancer and a static placement approach. The results show that at high levels of utilization, the difference between the algorithms' results is not so significant, but at lower utilization, the proposed algorithms perform significantly better, with pMaP being the best. Finally, it is important to note that the paper contains several other aspects beyond the algorithmic part, such as the pMapper architecture and practical experience about (deficiencies of) the performance isolation provided by virtualization.

6.2. Static Placement in a Single DC

A closely related paper, also from the IBM India Research Lab, investigates the opportunities for static placement in more detail. The work of Verma et al. [2009] starts with a very detailed empirical assessment of server traces from a real DC. Among other findings, they establish that VMs' actual resource requirements are most of the time less than half of the maximum value and that there is significant correlation

between the load of some pairs of VMs—especially those that belong to the same application. The authors then use these insights for designing optimization algorithms with the aim of minimizing energy consumption and the number of PM overloads, taking into account the single dimension of CPU load. The authors argue that for static (long-term) placement, the correlation between the loads of VMs, especially between their load peaks, is key: VMs with correlating load peak should not be placed on the same PM. They propose two new algorithms. Correlation-Based Placement (CBP) is an extension of pMapper's placement algorithm, using some given percentile of the load distribution of a VM (e.g., the size at 90% of the cumulative distribution function) as its size and avoiding the co-location of VMs with a correlation of their loads higher than a given limit. The other algorithm (Peak Clustering-Based Placement (PCP)) is completely new and works by clustering the VMs based on correlation between their load peaks and distributing VMs of the same cluster evenly among the PMs. The algorithms were evaluated using simulation, based on server traces from the DC where the first experiments were carried out. A comparison with pMapper's placement algorithm (which is optimized for dynamic placement) shows the superiority of the newly proposed algorithms for static placement, with PCP performing best in most cases. Finally, the authors show how the two new algorithms can be tuned, as they are quite sensitive to the used cutoff parameters and the training period.

6.3. Other Variants of the Single-DC Problem

Another paper that also starts with empirical investigations is the work of Srikantaiah (Pennsylvania State University) and Kansal and Zhao (Microsoft Research). Their focus is on the minimization of energy consumption by means of consolidation, subject to performance constraints [Srikantaiah et al. 2009]. In contrast to the works described earlier, they consider two kinds of resources: CPU and disk. The main finding of the paper is the observation that consolidation impacts performance and energy consumption in a highly nontrivial manner. Up to some point, increasing the utilization leads to higher energy efficiency, as expected. However, at some point, some resource of the PM saturates, and thus further increase in the utilization leads to performance degradation; since jobs take longer to complete, the energy consumption per job increases. As a result, energy consumption per job is a U-shaped function of utilization, yielding an optimal level of utilization. The authors propose to aim for this optimal utilization, which should be determined in an offline profiling phase. Afterward, a two-dimensional packing heuristic is used, where the bin sizes correspond to the optimal utilization of the PMs. The heuristic is a variation of WF, aiming at maximizing the remaining free capacity of PMs. This heuristic can be used both for accommodating new VMs and for optimizing the current placement of the VMs.

Multidimensional optimization of VM placement was also the subject of the work of Xiao et al. [2013] from Peking University. Their approach works in four steps: load prediction, hot spot elimination, cold spot elimination, and execution of migrations. For load prediction, an exponentially weighted average of past observations is used; however, weights are different for increasing and decreasing values so that the method reacts quickly if the load is increasing. Hot spots (PMs with load above some threshold in at least one dimension) are handled by greedily choosing VMs to migrate away from them. Cold spots (PMs with load below some threshold in each dimension) are handled only if the the average load of all PMs is below some given threshold; in that case, the algorithm tries to find a new host for the VMs on cold spot PMs; if a PM thus becomes empty, it can be switched off. In both hot spot and cold spot elimination, the *skewness* of the PMs is considered: this metric captures how unbalanced the resource load of the PM in the different dimensions is; the algorithm tries to minimize the skewness of the PMs. The proposed algorithm has been tested using both trace-based

simulation and real servers. The results demonstrate that the algorithm is very fast and—if the parameters are configured properly—effective in eliminating overloads and consolidating servers.

A quite different problem formulation was addressed by Meng et al. [2010] from IBM Research: they also consider the single-DC problem, but with the aim of minimizing network communication costs. The PMs' resources are not considered in detail, but it is assumed that some capacity planning approach has been used to define a number of *slots* on each PM, and the task is to map the VMs to the slots under the assumption that each VM fits into any slot. As input, the communication intensity is given for all pairs of VMs as well as the communication cost for all pairs of slots. The VM-to-slot mapping should minimize the resulting total communication cost. The authors classify this problem as a quadratic assignment problem and prove its NP-hardness by reduction from Balanced Minimum k-Cut. They propose a multilevel clustering algorithm: it clusters the VMs based on communication intensity, it also clusters the slots based on communication costs, it maps VM clusters to slot clusters, and then calls itself recursively for each of the generated VM cluster—slot cluster pairs. The next part of the paper is quite uncommon: for two special communication matrices and four network topologies, the authors try to determine the optimal cost (or, if this is not successful, a lower bound) and the expected cost of random placement to assess the optimization opportunities. The authors also evaluate the practical performance of their proposed algorithm and compare it to two general-purpose quadratic assignment heuristics using a combination of real server traces and synthetic additions. The results show that the proposed algorithm finds slightly better results with significantly shorter running time than the other heuristics.

6.4. Multi-IaaS Allocation

Communication costs also play a vital role in the DAG scheduling approaches that are common for multi-IaaS problem formulations. A representative example is the work of Pandey et al. [2010] from the University of Melbourne. Here, the aim is to map the tasks of a scientific workflow on cloud resources. For each task and each compute resource, it is given how long it would take and how much it would cost to execute the task on the resource. Dependencies between the tasks are given in the form of a DAG. For each edge of the DAG, the amount of transferred data is given; similarly, for each pair of compute resources, the cost of communication between them is given. For a mapping of tasks to resources, the total cost of a resource is defined as the execution cost of the task on this resource plus the sum of the data access costs along the incident edges; the objective is to minimize the maximum cost of a resource. For this optimization problem, the authors propose the use of Particle Swarm Optimization (PSO), a popular metaheuristic. Each particle encodes a task-resource mapping. The optimization is carried out in an online manner: first, the source tasks of the DAG are allocated; when some tasks have finished executing, the allocation of the tasks that are ready to be executed is again optimized using PSO and so on. The algorithm was evaluated using simulation on a rather small problem instance (three compute resources, five tasks) and compared to an algorithm that always selects the fastest but most expensive resource. Unsurprisingly, the solution found by the proposed algorithm incurs lower costs.

Another approach to the multi-IaaS problem is presented by Tordsson (Umea University) and Montero, Moreno-Vozmediano, and Llorente (Universidad Complutense de Madrid) [Tordsson et al. 2012]. Here we can select from a list of possible VM types, where each VM type is associated with a capacity. There are multiple CPs, and for each pair of VM type and CP, the hourly rental fee is given. The aim is to select a set of altogether n VMs from the CPs such that the total price is below a given limit and the total

capacity is maximal. This optimization problem is formulated as an integer program and solved using CPLEX, a commercial off-the-shelf solver. The authors also show how some further constraints can be incorporated (e.g., the number of VMs of a given type can be constrained). The algorithm is evaluated using three real CPs and four VM types, allocating a total of 16 VMs to run a distributed benchmark application. Experimenting with different cost limits, an interesting observation is that in most cases the optimal allocation involves more than one CP, highlighting the practicality of such a multicloud setup. Beyond the algorithmic part, the authors also discuss some other aspects, such as the role of a cloud broker in ensuring interoperability between different CPs.

7. AREAS IN NEED OF FURTHER RESEARCH

Despite the large amount of work already presented in the literature, there are still several aspects that, in our opinion, have not yet been addressed satisfactorily. This is true both for problem formulations and for algorithms. Moreover, the state of the art concerning the evaluation of algorithms also needs improvement. We elaborate on these topics next.

7.1. Problem Formulations

We see the following issues as the most important deficits in the prevalent problem formulations:

—*Hybrid cloud*: As mentioned previously, most works address either the single-DC or multi-IaaS problem. Very few works address hybrid clouds, and even those usually have a strong bias in the modeling either toward the single-DC or multi-IaaS subproblem, modeling the other parts only rudimentarily. Yet hybrid clouds play an increasingly important role in practice [Bittencourt et al. 2012a; Moreno-Vozmediano et al. 2011]. Especially in enterprise environments, hybrid clouds are becoming the standard, and so enterprise IT executives face decisions every day that relate to both in-house and cloud resources [Capgemini 2013]. Hence, in the future, we expect to see more research about genuine hybrid cloud problem formulations.
—*Task–VM–PM mapping*: In the single-DC problem, the usual formulation is about mapping VMs to PMs. In the multi-IaaS problem, it is more common to investigate the mapping of tasks to VMs. However, these are just two sides of the same coin: users actually want to get their tasks mapped to PMs, and VMs are just a tool that is used to enable this mapping in a safe and efficient way. This becomes especially clear in a hybrid cloud setting, where the users' tasks either are wrapped into VMs assigned to local PMs or are directly mapped to eCPs' VMs. Hence, in the future, we expect to see a converged model of the trilateral task–VM–PM assignment.
—*Co-optimization*: VM placement is just one level where power consumption is optimized. But power consumption optimization techniques are also implemented on the server level (e.g., DVFS), the level of individual components (switching unused cores, cache ways, memory banks, disks, etc., to a low-energy state), and in network equipment (routers, switches), altogether making up a very complex system. In particular, it is not clear how these different optimization techniques interact and possibly interfere with each other [Raghavendra et al. 2008]. It is not clear whether the optimal decision in the VM placement problem, if it does not account for the other optimization levels, is indeed the best choice for the overall system's power consumption. More research is needed to better understand these interactions.
—*Multicore CPUs*: The existing problem formulations in the literature either do not model multicore CPUs at all or model them in a very simplistic way. This

compromises the practical applicability of such approaches, because multicore CPUs are now omnipresent.

—*Communication*: Data transfer among VMs is another aspect severely missing from many existing problem formulations, especially in the case of the single-DC problem, although it can impact overall system performance substantially. In the literature about the multi-IaaS problem, communication among tasks is more frequently taken into account, yet almost exclusively coupled with the assumption that all dependencies are given in the form of a DAG. However, in practice, there are often cyclic communication scenarios (e.g., two applications regularly exchanging information in both directions), and in many cases the communication paths are not static but change at runtime depending on real-time information. In addition, the modeling of workflows as DAGs usually assumes finish-to-start dependencies between adjacent tasks, but in practice the second task can usually start its execution once some partial results of the first task are available. For these reasons, although the DAG scheduling approach is tempting because of its theoretical clarity, its applicability is limited to some narrow domains. Further research is needed on more widely applicable models of communication among VMs.

—*Co-location interference*: When deciding to place a set of VMs on a PM, many works only check that the total size of the VMs does not exceed the PM's capacity. However, in practice, there are also other attributes of the VMs that influence how suitable they are for co-location. One factor to consider is correlation: how likely it is that the resource demand of several of the VMs will increase at the same time [Verma et al. 2009]. Another factor is the "noisy neighbor" effect: since current virtualization technologies do not offer complete performance isolation of the co-located VMs, if one of the VMs uses a resource excessively, this may degrade the performance of the others [Kim et al. 2013]. Up to now, very few works addressed these issues.

7.2. Algorithms

The most important deficiency in terms of algorithms is that mostly heuristic algorithms (in most cases, quite simple heuristics) have been proposed, without any performance guarantee. This is problematic because even if they perform well in controlled experiments, they may yield poor results in real settings, especially for large and highly constrained problem instances.

—*Exact algorithms*: Since the VM placement problem contains the bin-packing problem as special case, which is NP-hard in the strong sense [Martello and Toth 1990], there is no hope for an exact algorithm with polynomial or even pseudopolynomial runtime. Nevertheless, there is still much that could be done in the context of exact algorithms—for example, efficiently solvable special cases, fixed-parameter tractability, randomized algorithms with limited error probability, and algorithms with low typical-case complexity [Mann 2011]. Those authors who did experiment with exact solutions usually used *off-the-shelf solvers* for different classes of mathematical programming; the fact that those solvers took a long time to solve even mid-sized problem instances does not mean that it is not possible to come up with better exact algorithms tailored specifically to the given problem.

—*Approximation algorithms*: Another logical possibility that has largely been unexplored would be to use approximation algorithms, i.e., polynomial-time algorithms that are guaranteed to deliver a result with cost at most constant times the optimum. Since there are good approximation algorithms for the bin-packing problem, this may suggest that similar results could also be achieved for the VM placement problem.

—*Coping with uncertainty*: Most algorithms assume that all parameters of the problem are fixed and precisely known constants. (Even the approaches that attack the load prediction problem assume that parameters other than the VMs' load are fixed and precisely known.) However, in reality, cloud DCs are very complex and highly dynamic systems, so a real cloud management system must cope with estimation errors (e.g., PMs' background load is not constant, so the estimate of a PM's available capacity may turn out be incorrect) and unforeseen events (e.g., a PM may be damaged or may need to be restarted because of an urgent operating system patch). The algorithms presented so far in the literature are usually not robust enough to handle such situations.

7.3. Evaluation of Algorithms

Besides the problem formulations and the proposed algorithms, we feel a need for improvement in the way in which algorithms for VM allocation are usually evaluated:

—*Analytic evaluation*: Most papers in the literature completely lack an analytic evaluation of the proposed algorithms. As a minimum, an estimation of the asymptotic worst-case runtime and memory consumption of the algorithms should be given. If mathematically feasible, an estimation of the asymptotic *average-case behavior* of the algorithms (using some appropriate probability distribution of the input parameters) would be even more interesting. Alternatively, an analysis of some more easily handled special cases would also contribute to a better understanding of, and thus to an increased confidence in, the proposed algorithms.

—*Empirical evaluation*: In absence of a detailed analytic evaluation, the empirical evaluation of the proposed algorithms is very important. Ideally, each new paper should show the advantages of the proposed method by means of a systematic comparison to previously suggested methods on a large number of different, practically relevant benchmark instances. Unfortunately, this is hardly ever done. One problem is that there are no widely accepted benchmarks for the VM placement problem (and its special cases), and another issue is the co-existence of many different problem formulations, making meaningful comparisons difficult. But independently of these issues, researchers often compare their approaches to trivial algorithms or to algorithms that do not take into account some important characteristic of the problem, compare different versions of their own algorithm to each other, or do not do any comparison at all. As a result, at this time, we have no way to tell which of the proposed algorithms works best. The community will need to develop more rigor concerning the empirical evaluation of algorithms to better support future development of the field.

8. CONCLUSIONS

We have presented a survey of the state of the art in the VM allocation problem concerning problem models and algorithmic approaches. Because of the large number of papers in this field, we could not describe all of them, but we tried to show a representative selection of the most important works. As we have seen, most papers deal with either the single-DC or multi-IaaS problem, but also within those two big clusters, there are significant differences between the problem formulations used in each paper. Currently, the literature on these two subproblems is mostly disjoint, with only few works addressing a combination of the two. However, we argued that to capture hybrid cloud scenarios, a convergence of these two fields will be necessary in the future.

Given the diversity of the available approaches to VM placement, natural questions that arise are which method is best—and, more realistically, when to use which method. Unfortunately, the heterogeneity of the considered problem formulations and the lack

of meaningful algorithm comparison studies make it very hard to answer these questions. We definitely see the need for future work comparing the real-world performance of algorithms under different scenarios. In addition, a regular competition would be very helpful for the community, similarly to competitions in other fields, such as the Competition on Software Verification (http://sv-comp.sosy-lab.org/).

For now, we can make recommendations mainly based on problem formulations. In other words, to find out which approaches may be most suitable in a given situation, one should first determine if the single-DC, the multi-IaaS, or some of the other variants apply. Then, the main characteristics should be identified according to Table I or Table II. For example, in the case of communication-intensive workloads, one should consult the approaches that take inter-VM communication costs into account; likewise, if there are stringent SLAs on response time, then one should focus on approaches that support such user-level SLOs and so forth. This way, the search can be narrowed down to a small number of works that need to be evaluated in detail.

It is our hope that our survey will help practitioners select the most appropriate existing works and that it will contribute to the maturation of this important and challenging field by demonstrating both previous achievements and areas for future research.

REFERENCES

Abdulla M. Al-Qawasmeh, Sudeep Pasricha, Anthony A. Maciejewski, and Howard Jay Siegel. 2015. Power and thermal-aware workload allocation in heterogeneous data centers. *IEEE Transactions on Computers* 64, 2, 477–491.

Mansoor Alicherry and T. V. Lakshman. 2012. Network aware resource allocation in distributed clouds. In *Proceedings of IEEE Infocom*. 963–971.

Mansoor Alicherry and T. V. Lakshman. 2013. Optimizing data access latencies in cloud systems by intelligent virtual machine placement. In *Proceedings of IEEE Infocom*. 647–655.

Amid Khatibi Bardsiri and Seyyed Mohsen Hashemi. 2012. A review of workflow scheduling in cloud computing environment. *International Journal of Computer Science and Management Research* 1, 3, 348–351.

Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. 2013. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* (2nd ed.). Morgan and Claypool.

Daniel M. Batista, Nelson L. S. da Fonseca, and Flavio K. Miyazawa. 2007. A set of schedulers for grid networks. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC'07)*. 209–213.

Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems* 28, 755–768.

Anton Beloglazov and Rajkumar Buyya. 2010a. Energy efficient allocation of virtual machines in cloud data centers. In *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing*. 577–578.

Anton Beloglazov and Rajkumar Buyya. 2010b. Energy efficient resource management in virtualized cloud data centers. In *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing*. 826–831.

Anton Beloglazov and Rajkumar Buyya. 2012. 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.

Anton Beloglazov and Rajkumar Buyya. 2013. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems* 24, 7, 1366–1379.

Ofer Biran, Antonio Corradi, Mario Fanelli, Luca Foschini, Alexander Nus, Danny Raz, and Ezra Silvera. 2012. A stable network-aware VM placement for cloud systems. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. IEEE, Los Alamitos, CA, 498–506.

Luiz F. Bittencourt and Edmundo R. M. Madeira. 2011. HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications* 2, 3, 207–227.

Luiz F. Bittencourt, Edmundo R. M. Madeira, and Nelson L. S. da Fonseca. 2012a. Scheduling in hybrid clouds. *IEEE Communications Magazine* 50, 9, 42–47.

Luiz F. Bittencourt, Rizos Sakellariou, and Edmundo R. M. Madeira. 2012b. Using relative costs in workflow scheduling to cope with input data uncertainty. In *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds, and e-Science*. Article No. 8.

Norman Bobroff, Andrzej Kochut, and Kirk Beaty. 2007. Dynamic placement of virtual machines for managing SLA violations. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*. 119–128.

Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. 2010. Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing*. 228–235.

David Breitgand and Amir Epstein. 2011. SLA-aware placement of multi-virtual machine elastic services in compute clouds. In *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management*. 161–168.

David Breitgand and Amir Epstein. 2012. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In *Proceedings of IEEE Infocom*. 2861–2865.

Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. 2010. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. In *Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications*. 6–17.

Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 6, 599–616.

Rodrigo N. Calheiros and Rajkumar Buyya. 2014. Meeting deadlines of scientific workflows in public clouds with tasks replication. *IEEE Transactions on Parallel and Distributed Systems* 25, 7, 1787–1796.

Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1, 23–50.

David Candeia, Ricardo Araújo, Raquel Lopes, and Francisco Brasileiro. 2010. Investigating business-driven cloudburst schedulers for e-science bag-of-tasks applications. In *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science*. 343–350.

Capgemini. 2013. Simply. Business Cloud. Retrieved July 14, 2015, from http://www.capgemini.com/resource-file-access/resource/pdf/simply._busi ness_cloud_where_business_meets_cloud.pdf.

Emiliano Casalicchio, Daniel A. Menascé, and Arwa Aldhalaan. 2013. Autonomic resource provisioning in cloud systems with availability goals. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*. Article No. 1.

Emmanuel Cecchet, Anupam Chanda, Sameh Elnikety, Julie Marguerite, and Willy Zwaenepoel. 2003. Performance comparison of middleware architectures for generating dynamic Web content. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. 242–261.

Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. 2009. Optimal virtual machine placement across multiple cloud providers. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC'09)*. 103–110.

Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, and Amin M. Vahdat. 2001. Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*. 103–116.

Yuan Chen, Subu Iyer, Xue Liu, Dejan Milojicic, and Akhil Sahai. 2008. Translating service level objectives to lower level policies for multi-tier services. *Cluster Computing* 11, 299–311.

Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. 2007. *When Virtual Is Harder Than Real: Resource Allocation Challenges in Virtual Machine Based IT Environments*. Technical Report. HP Laboratories, Palo Alto, CA.

Navraj Chohan, Claris Castillo, Mike Spreitzer, Malgorzata Steinder, Asser Tantawi, and Chandra Krintz. 2011. See spot run: Using spot instances for MapReduce workflows. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10)*. 7.

Rajarshi Das, Jeffrey O. Kephart, Charles Lefurgy, Gerald Tesauro, David W. Levine, and Hoi Chan. 2008. Autonomic multi-agent management of power and performance in data centers. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*. 107–114.

Digital Power Group. 2013. The Cloud Begins with Coal—Big Data, Big Networks, Big Infrastructure, and Big Power. Retrieved July 14, 2015, from http://www.tech-pundit.com/wp-content/uploads/2013/07/Cloud_Begins_With_Coal.pdf.

Dinil Mon Divakaran, Tho Ngoc Le, and Mohan Gurusamy. 2014. An online integrated resource allocator for guaranteed performance in data centers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6, 1382–1392.

György Dósa. 2007. 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*. Springer, 1–11.

György Dósa and Jiří Sgall. 2013. First fit bin packing: A tight analysis. In *Proceedings of the 30th Symposium on Theoretical Aspects of Computer Science (STACS'13)*. 538–549.

György Dósa and Jiří Sgall. 2014. Optimal analysis of best fit bin packing. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*. 429–441.

Dror G. Feitelson, Dan Tsafrir, and David Krakov. 2014. Experience with using the parallel workloads archive. *Journal of Parallel and Distributed Computing* 74, 10, 2967–2982.

Ana Juan Ferrer, Francisco Hernández, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, Raül Sirvent, Jordi Guitart, Rosa M. Badia, Karim Djemame, Wolfgang Ziegler, Theo Dimitrakos, Srijith K. Nair, George Kousiouris, Kleopatra Konstanteli, Theodora Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesner, Marcelo Corrales, Nikolaus Forgó, Tabassum Sharif, and Craig Sheridan. 2012. OPTIMIS: A holistic approach to cloud service provisioning. *Future Generation Computer Systems* 28, 1, 66–77.

Yongqiang Gao, Haibing Guan, Zhengwei Qi, Yang Hou, and Liang Liu. 2013. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences* 79, 1230–1242.

Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. 2013. A framework for ranking of cloud computing services. *Future Generation Computer Systems* 29, 4, 1012–1023.

Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. 2012. Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels. In *Proceedings of the Network Operations and Management Symposium (NOMS'12)*. IEEE, Los Alamitos, CA, 906–912.

Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, Guillaume Belrose, Tom Turicchi, and Alfons Kemper. 2008. An integrated approach to resource pool management: Policies, efficiency and quality metrics. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks*. 326–335.

Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. 2009. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks* 53, 17, 2905–2922.

Marco Guazzone, Cosimo Anglano, and Massimo Canonico. 2012a. Exploiting VM migration for the automated power and performance management of green cloud computing systems. In *Proceedings of the 1st International Workshop on Energy Efficient Data Centers (E2DC'12)*. 81–92.

Marco Guazzone, Cosimo Anglano, and Massimo Canonico. 2012b. *Exploiting VM Migration for the Automated Power and Performance Management of Green Cloud Computing Systems*. Technical Report TR-INF-2012-04-02-UNIPMN. University of Piemonte Orientale.

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

Ahmad Fadzil M. Hani, Irving Vitra Paputungan, and Mohd Fadzil Hassan. 2015. Renegotiation in service level agreement management for a cloud-based system. *ACM Computing Surveys* 47, 3.

Sijin He, Li Guo, Moustafa Ghanem, and Yike Guo. 2012. Improving resource utilisation in the cloud environment using multivariate probabilistic models. In *Proceedings of the IEEE 5th International Conference on Cloud Computing*. 574–581.

Tibor Horvath, Tarek Abdelzaher, Kevin Skadron, and Xue Liu. 2007. Dynamic voltage scaling in multi-tier Web servers with end-to-end delay control. *IEEE Transactions on Computers* 56, 4, 444–458.

Chris Hyser, Bret McKee, Rob Gardner, and Brian J. Watson. 2008. *Autonomic Virtual Machine Placement in the Data Center*. Technical Report. HP Laboratories.

Waheed Iqbal, Matthew N. Dailey, and David Carrera. 2010. SLA-driven dynamic resource management for multi-tier Web applications in a cloud. In *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing (CCGrid'10)*. 832–837.

Deepal Jayasinghe, Calton Pu, Tamar Eilam, Malgorzata Steinder, Ian Whalley, and Ed Snible. 2011. Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement. In *Proceedings of the IEEE International Conference on Services Computing (SCC'11)*. 72–79.

Joe Wenjie Jiang, Tian Lan, Sangtae Ha, Minghua Chen, and Mung Chiang. 2012. Joint VM placement and routing for data center traffic engineering. In *Proceedings of IEEE Infocom*. 2876–2880.

Gueyoung Jung, Matti A. Hiltunen, Kaustubh R. Joshi, Richard D. Schlichting, and Calton Pu. 2010. Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In *Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS'10)*. 62–73.

Gabor Kecskemeti, Gabor Terstyanszky, Peter Kacsuk, and Zsolt Nemeth. 2011. An approach for virtual appliance distribution for service deployment. *Future Generation Computer Systems* 27, 3, 280–289.

Matthias Keller and Holger Karl. 2014. Response time-optimized distributed cloud resource allocation. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*. 47–52.

Gunjan Khanna, Kirk Beaty, Gautam Kar, and Andrzej Kochut. 2006. Application performance management in virtualized server environments. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium*. 373–381.

Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. 2013. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In *Proceedings of the 19th International Conference on Parallel Processing (Euro-Par'13)*. 317–328.

Shin-gyu Kim, Hyeonsang Eom, and Heon Y. Yeom. 2013. Virtual machine consolidation based on interference modeling. *Journal of Supercomputing* 66, 3, 1489–1506.

Ricardo Koller, Akshat Verma, and Anindya Neogi. 2010. WattApp: An application aware power meter for shared data centers. In *Proceedings of the 7th International Conference on Autonomic Computing*. 31–40.

Ricardo Koller, Akshat Verma, and Raju Rangaswami. 2011. Estimating application cache requirements for provisioning caches in virtualized systems. In *Proceedings of the IEEE 19th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'11)*. 55–62.

Kleopatra Konstanteli, Tommaso Cucinotta, Konstantinos Psychas, and Theodora A. Varvarigou. 2014. Elastic admission control for federated cloud services. *IEEE Transactions on Cloud Computing* 2, 3, 348–361.

Madhukar Korupolu, Aameek Singh, and Bhuvan Bamba. 2009. Coupled placement in modern data centers. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS'09)*. 1–12.

Daniel Guimaraes do Lago, Edmundo R. M. Madeira, and Luiz Fernando Bittencourt. 2011. Power-aware virtual machine scheduling on clouds using active cooling control and DVFS. In *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds, and e-Science*. Article No. 2.

Ulrich Lampe, Melanie Siebenhaar, Ronny Hans, Dieter Schuller, and Ralf Steinmetz. 2012. Let the clouds compute: Cost-efficient workload distribution in infrastructure clouds. In *Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON'12)*. 91–101.

Wubin Li, Petter Svärd, Johan Tordsson, and Erik Elmroth. 2012. A general approach to service deployment in cloud environments. In *Proceedings of the 2nd International Conference on Cloud and Green Computing (CGC'12)*. 17–24.

Wubin Li, Petter Svärd, Johan Tordsson, and Erik Elmroth. 2013. Cost-optimal cloud service placement under dynamic pricing schemes. In *Proceedings of the 6th IEEE/ACM International Conference on Utility and Cloud Computing*. 187–194.

Wubin Li, Johan Tordsson, and Erik Elmroth. 2011a. Modeling for dynamic cloud scheduling via migration of virtual machines. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science*. 163–171.

Wubin Li, Johan Tordsson, and Erik Elmroth. 2011b. 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'11)*. 120–134.

Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. 2009. GreenCloud: A new architecture for green data center. In *Proceedings of the 6th International Conference on Autonomic Computing and Communications*. 29–38.

Zoltán Ádám Mann. 2011. *Optimization in Computer Engineering—Theory and Applications*. Scientific Research Publishing.

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

Zoltán Ádám Mann. 2015b. Modeling the virtual machine allocation problem. In *Proceedings of the International Conference on Mathematical Methods, Mathematical Models, and Simulation in Science and Engineering*. 102–106.

Zoltán Ádám Mann. 2015c. 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.

Silvano Martello and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and Sons.

Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. 2010. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings of IEEE INFOCOM*. 1–9.

Kevin Mills, James Filliben, and Christopher Dabrowski. 2011. Comparing VM-placement algorithms for on-demand clouds. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science*. 91–98.

Mayank Mishra and Anirudha Sahoo. 2011. On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *Proceedings of the IEEE International Conference on Cloud Computing*. 275–282.

Christoph Mobius, Waltenegus Dargie, and Alexander Schill. 2014. Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6, 1600–1614.

Rafael Moreno-Vozmediano, Ruben S. Montero, and Ignacio M. Llorente. 2011. Multicloud deployment of computing clusters for loosely coupled MTC applications. *IEEE Transactions on Parallel and Distributed Systems* 22, 6, 924–930.

Ripal Nathuji and Karsten Schwan. 2007. VirtualPower: Coordinated power management in virtualized enterprise systems. In *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP'07)*. 265–278.

Daniel de Oliveira, Kary A. C. S. Ocana, Fernanda Baiao, and Marta Mattoso. 2012. A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *Journal of Grid Computing* 10, 521–552.

Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA'10)*. IEEE, Los Alamitos, CA, 400–407.

Christos H. Papadimitriou and Mihalis Yannakakis. 2001. Multiobjective query optimization. In *Proceedings of the 20th ACM Symposium on Principles of Database Systems*. ACM, New York, NY, 52–59.

Michael L. Pinedo. 2008. *Scheduling: Theory, Algorithms, and Systems* (3rd ed.). Springer.

Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. 2001. Load balancing and unbalancing for power and performance in cluster-based systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*. 182–195.

Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. 2008. No "power" struggles: Coordinated multi-level power management for the data center. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*. 48–59.

Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch. 2012. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC'12)*. Article No. 7.

Bruno Cesar Ribas, Rubens Massayuki Suguimoto, Razer A. N. R. Montano, Fabiano Silva, Luis de Bona, and Marcos A. Castilho. 2012. On modelling virtual machine consolidation to pseudo-Boolean constraints. In *Proceedings of the 13th Ibero-American Conference on AI*. 361–370.

Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. 2008. A comparison of high-level full-system power models. In *Proceedings of the Workshop on Power Aware Computing and Systems (HotPower'08)*. 3.

Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Llorente, Ruben Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, M. Ben-Yehuda, Wolfgang Emmerich, and Fermin Galán. 2009. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development* 53, 4, 1–11.

Ivan Rodero, Hariharasudhan Viswanathan, Eun Kyung Lee, Marc Gamell, Dario Pompili, and Manish Parashar. 2012. Energy-efficient thermal-aware autonomic management of virtualized HPC cloud infrastructure. *Journal of Grid Computing* 10, 3, 447–473.

Mohsen Amini Salehi, P. Radha Krishna, Krishnamurty Sai Deepak, and Rajkumar Buyya. 2012. Preemption-aware energy management in virtualized data centers. In *Proceedings of the 5th International Conference on Cloud Computing*. IEEE, Los Alamitos, CA, 844–851.

Mina Sedaghat, Francisco Hernandez-Rodriguez, and Erik Elmroth. 2013. A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*. Article No. 6.

Damián Serrano, Sara Bouchenak, Yousri Kouki, Frederico Alvares de Oliveira Jr., Thomas Ledoux, Jonathan Lejeune, Julien Sopena, Luciana Arantes, and Pierre Sens. 2015. SLA guarantees for cloud services. *Future Generation Computer Systems*. In Press.

Lei Shi, John Furlong, and Runxin Wang. 2013. Empirical evaluation of vector bin packing algorithms for energy efficient data centers. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'13)*. 9–15.

Weijia Song, Zhen Xiao, Qi Chen, and Haipeng Luo. 2014. Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers* 63, 11, 2647–2660.

Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. 2009. Energy aware consolidation for cloud computing. *Cluster Computing* 12, 1–15.

Steve Strauch, Oliver Kopp, Frank Leymann, and Tobias Unger. 2011. A taxonomy for cloud data hosting solutions. In *Proceedings of the IEEE International Conference on Cloud and Green Computing (CGC'11)*. 577–584.

Anja Strunk. 2012. Costs of virtual machine live migration: A survey. In *Proceedings of the 8th IEEE World Congress on Services*. 323–329.

Rick Sturm and Wayne Morris. 2000. *Foundations of Service Level Management*. SAMS Publishing.

Petter Svärd, Wubin Li, Eddie Wadbro, Johan Tordsson, and Erik Elmroth. 2014. *Continuous Datacenter Consolidation*. Technical Report. Umea University.

Vanish Talwar, Dejan Milojicic, Qinyi Wu, Calton Pu, Wenchang Yan, and Gueyoung Jung. 2005. Approaches for service deployment. *IEEE Internet Computing* 9, 2, 70–80.

Selome K. Tesfatsion, Eddie Wadbro, and Johan Tordsson. 2014. A combined frequency scaling and application elasticity approach for energy-efficient cloud computing. *Sustainable Computing: Informatics and Systems* 4, 4, 205–214.

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

Johan Tordsson, Rubén S. Montero, Rafael Moreno-Vozmediano, and Ignacio M. Llorente. 2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems* 28, 2, 358–367.

Efthymia Tsamoura, Anastasios Gounaris, and Kostas Tsichlas. 2013. Multi-objective optimization of data flows in a multi-cloud environment. In *Proceedings of the 2nd Workshop on Data Analytics in the Cloud*. 6–10.

Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, and Pawan Goyal. 2005. Dynamic provisioning of multi-tier Internet applications. In *Proceedings of the 2nd International Conference on Autonomic Computing*. 217–228.

Akshat Verma, Puneet Ahuja, and Anindya Neogi. 2008a. pMapper: Power and migration cost aware application placement in virtualized systems. In *Proceedings of Middleware 2008*. 243–264.

Akshat Verma, Puneet Ahuja, and Anindya Neogi. 2008b. Power-aware dynamic placement of HPC applications. In *Proceedings of the 22nd Annual International Conference on Supercomputing*. 175–184.

Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari. 2009. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 USENIX Annual Technical Conference*. 355–368.

Akshat Verma, Gautam Kumar, and Ricardo Koller. 2010. The cost of reconfiguration in a cloud. In *Proceedings of the 11th International Middleware Conference*. 11–16.

Akshat Verma, Gautam Kumar, Ricardo Koller, and Aritra Sen. 2011. CosMig: Modeling the impact of reconfiguration in a cloud. In *Proceedings of the 19th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'11)*. IEEE, Los Alamitos, CA, 3–11.

David Villegas, Athanasion Antoniou, Seyed Masoud Sadjadi, and Alexandru Iosup. 2012. An analysis of provisioning and allocation policies for Infrastructure-as-a-Service clouds. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid'12)*. 612–619.

Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. 2009. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks* 53, 17, 2923–2938.

Zhangjun Wu, Zhiwei Ni, Lichuan Gu, and Xiao Liu. 2010. A revised discrete particle swarm optimization for cloud workflow scheduling. In *Proceedings of the International Conference on Computational Intelligence and Security*. 184–188.

Zhen Xiao, Qi Chen, and Haipeng Luo. 2014. Automatic scaling of Internet applications for cloud computing services. *IEEE Transactions on Computers* 63, 5, 1111–1123.

Zhen Xiao, Weijia Song, and Qi Chen. 2013. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems* 24, 6, 1107–1117.

Pengcheng Xiong, Yun Chi, Shenghuo Zhu, Hyun Jin Moon, Calton Pu, and Hakan Hacgumus. 2015. SmartSLA: Cost-sensitive management of virtualized resources for CPU-bound database services. *IEEE Transactions on Parallel and Distributed Systems* 26, 5, 1441–1451.

Jing Xu and Jose A. B. Fortes. 2010. Multi-objective virtual machine placement in virtualized data center environments. In *Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications and the International Conference on Cyber, Physical, and Social Computing (GREENCOM-CPSCOM'10)*. 179–188.

Qi Zhang, Lu Cheng, and Raouf Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1, 1, 7–18.

Xiaoyun Zhu, Donald Young, Brian J. Watson, Zhikui Wang, Jerry Rolia, Sharad Singhal, Bret McKee, Chris Hyser, Daniel Gmach, Robert Gardner, Tom Christian, and Ludmila Cherkasova. 2009. 1000 islands: An integrated approach to resource management for virtualized data centers. *Cluster Computing* 12, 1, 45–57.

Received August 2014; revised May 2015; accepted June 2015