FOCUS

Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters

Luca Caviglione[1] · Mauro Gaggero[2] · Massimo Paolucci[3] · Roberto Ronco[3]

Abstract

The ubiquitous diffusion of cloud computing requires suitable management policies to face the workload while guaranteeing quality constraints and mitigating costs. The typical trade-off is between the used power and the adherence to a service-level metric subscribed by customers. To this aim, a possible idea is to use an optimization-based placement mechanism to select the servers where to deploy virtual machines. Unfortunately, high packing factors could lead to performance and security issues, e.g., virtual machines can compete for hardware resources or collude to leak data. Therefore, we introduce a multi-objective approach to compute optimal placement strategies considering different goals, such as the impact of hardware outages, the power required by the datacenter, and the performance perceived by users. Placement strategies are found by using a deep reinforcement learning framework to select the best placement heuristic for each virtual machine composing the workload. Results indicate that our method outperforms bin packing heuristics widely used in the literature when considering either synthetic or real workloads.

Keywords Deep reinforcement learning · Multi-objective optimization · Virtual machine placement

1 Introduction

The cloud paradigm was originally introduced to access computing resources on demand, and nowadays it has become the foundation of different services. Servers within a datacenter are used to deliver applications and stream contents to an Internet-wide user population, host and process big-data-like information, or provide computational assets to other layers located within telcos and industrial settings. This trend culminates in the Everything-as-a-Service approach, which continues to grow and also accounts for issues in terms of security, privacy and confidentiality, economical costs, energetic usages, and environmental footprints (Duan et al. 2015; Gaggero and Caviglione 2019). As a consequence, modern cloud datacenters should be able to satisfy a variety of performance goals. For instance, users should not perceive degradation of the subscribed service-level agreement (SLA). At the same time, the owner of the computing infrastructure aims at minimizing capital and operating expenditures, e.g., in terms of hardware and power usage. Unfortunately, such goals are often conflicting, as costs are primarily tamed by reducing the needed resources, which could lead to overloaded servers or multiple processes sharing the same hardware, thus reducing overall efficiency.

In this perspective, the literature abounds in works proposing techniques for the optimization of cloud datacenters. A possible approach exploits the ability of CPUs to vary their working frequency at run-time to match the workload (Guenter et al. 2011). However, this prevents from taking advantage of the flexibility offered by cloud frameworks built

✉ Massimo Paolucci
massimo.paolucci@unige.it

Luca Caviglione
luca.caviglione@ge.imati.cnr.it

Mauro Gaggero
mauro.gaggero@cnr.it

Roberto Ronco
roberto.ronco@edu.unige.it

[1] IMATI, National Research Council of Italy, Via De Marini 6, 16149 Genoa, Italy

[2] INM, National Research Council of Italy, Via De Marini 6, 16149 Genoa, Italy

[3] DIBRIS, University of Genoa, Via All'Opera Pia 13, 16145 Genoa, Italy

on top of servers partitioned into virtual machines (VMs) and from the use of middleware layers to control access to resources (Ghobaei-Arani et al. 2019). Hence, to pursue the vision of efficient cloud datacenters, the preferred solution aims at finding the best mapping of VMs over servers or other computing resources, referred to as physical machines (PMs) in the following, according to some performance criteria. In general, two approaches can be adopted (Gaggero and Caviglione 2019). The first one is called consolidation and allows to periodically rearrange VMs to match or compensate fluctuations of the workload (see Xu et al. 2017 for a survey) or to pursue a trade-off between energy and quality metrics (Li et al. 2020). However, as detailed in Zhang et al. (2018), live migration of VMs poses various technological and performance challenges, and it requires non-negligible network bandwidth and computing resources. For this reason, another approach called placement has been developed to prevent inefficient allocations when VMs are firstly created on PMs to fulfill requests from users (Usmani and Singh 2016).

The use of optimized placement mechanisms proved to be successful in a broad set of use cases, including production-quality scenarios (Ahmad et al. 2015). A typical solution exploits heuristics based on bin packing (Panigrahy et al. 2011). In fact, VM placement can be modeled as a bin-packing problem, where VMs and PMs are objects and bins, respectively. For instance, the *first fit* heuristic allows to place VMs over PMs in an efficient manner, but at the price of too aggressive packings causing VMs to interfere with each others. More refined approaches like *dot product* and *norm2* can be used to reduce such a drawback, as they weight different performance metrics when computing the VM-to-PM mapping (e.g., the energetic footprints vs. the amount of resources delivered to users) (Song et al. 2013; Srikantaiah et al. 2008). Yet, in the presence of multiple and conflicting goals, heuristics based on bin packing principles could be inefficient. Thus, refinements have been proposed to endow best-fit algorithms with prediction capabilities to explicitly consider energy efficiency and SLA violations (Ghobaei-Arani et al. 2017, 2018). Other solutions exploit optimization to guarantee a more fine-grained control over the trade-off between placement actions and performance objectives, e.g., used power, reliability of the hardware, and mitigation of information leakage between VMs (Gaggero and Caviglione 2019; Caviglione and Gaggero 2021).

Summarizing, VM placement is an interplay of different objectives, constraints, and technological domains. Machine learning techniques can tame such a complexity, owing to their capability of finding "hidden" relationships among the available data and therefore generate placement actions that may be difficult to be found using classical optimization tools or heuristics based on common sense. Machine learning can be used either to design new VM placement approaches or to enhance the capabilities of existing heuristics. Toward this

end, in this paper we propose a mechanism for VM placement based on deep reinforcement learning (DRL) (Arulkumaran et al. 2017). Specifically, we consider a decision maker that, after a proper training, is able to select the most suitable heuristic to compute the placement for each VM requested by end users.

To the best of our knowledge, this paper is the first one using DRL to implement a decision maker solving a multi-objective VM placement problem. This novel multi-objective approach represents the main contribution of the paper. To evaluate its effectiveness, comparisons against solutions widely adopted in the literature and real-world scenarios are presented, including the use of workload traces collected in a production-quality cloud datacenter.

The rest of this paper is structured as follows. Section 2 reviews the literature on placement techniques with emphasis on those leveraging machine learning. Section 3 formalizes VM placement as a multi-objective combinatorial problem. Section 4 discusses the proposed DRL-based approach. Section 5 presents numerical results obtained via a simulation campaign. Section 6 concludes the paper and showcases possible future research directions.

2 Related work

Despite using consolidation, placement, or a combination of both techniques to enforce the performances of Infrastructure-as-a-Service (IaaS) cloud datacenters, earlier works mainly focused on the search for a trade-off between the power needed to operate the hardware and the quality perceived by users (Kaur and Chana 2015). However, the complexity of virtualized environments and the progressive convergence of cloud with wireless and vehicular networks, as well as the need of supporting an Internet-wide user population while guaranteeing privacy and security requirements, impose not to limit the scope of optimization to energetic aspects, but also to jointly pursue several performance goals (Caviglione et al. 2017; Gaggero and Caviglione 2019; Caviglione and Gaggero 2021). Large-scale installations could also require to explicitly consider the internal network architecture, thus making the problem of finding suitable mappings between VMs and PMs more complex (Malekloo and Kara 2014).

Owing to the pervasive nature of IaaS technologies, many approaches taking advantage of a variety of techniques have been proposed in the last decade to address the VM placement problem (Masdari et al. 2016). For instance, we mention model predictive control to exploit future information inferred from the incoming workload of requests (Gaggero and Caviglione 2019), ant colony optimization to obtain Pareto-optimal solutions of multi-objective problems considering power, resource usage, and quality of service

requirements (Malekloo and Kara 2014; Gao et al. 2013), as well as bio-inspired methodologies (Masdari et al. 2019) and ad hoc heuristics to prevent that an inefficient use of the network bandwidth voids the applicability in real-world use cases (Farzai et al. 2020).

A recent trend exploits machine learning to develop scalable, proactive, and efficient frameworks to optimize IaaS deployments (Ismaeel et al. 2018). In this vein, Farahnakian et al. (2014) exploit Q-learning, a reinforcement learning (Sutton and Barto 2018) algorithm, to infer the power model of a server that can be put in the sleep state prior migrating the hosted VMs. Another effort to address the VM placement problem with reinforcement learning has been carried out in Qin et al. (2020), where Q-learning is used to optimize two conflicting objectives, while finding an approximation of the Pareto front. In more detail, the authors consider "continuous" CPU and memory resources, as opposed to our work, where we construct a framework based on the allocation of a fixed, limited number of VM classes. Moreover, we consider multiple conflicting objectives instead of only focusing on energy requirements and resource wastage. Unfortunately, this class of approaches is often affected by slow convergence. To face such issue, Shaw et al. (2017) propose to use an expert advice as part of the learning model to let the agent learn in a more rapid manner. A limit of Q-learning that prevents its adoption in large-scale IaaS deployments is the "curse of dimensionality," i.e., an exponential growth of the complexity with the size of the datacenter.

Another technique widely used for the management of complex computing and networking infrastructures is DRL, which overcomes the curse of dimensionality by using neural networks as function approximators. By enhancing reinforcement learning techniques with deep learning capabilities for representing the state of the datacenter, it is possible to tackle large instances of the VM placement problem. A significant example is given in Liu et al. (2017), where a method to schedule jobs while maximizing the number of machines that can be shut down to save power is presented. However, it does not take advantage of VMs since it considers the server as a monolithic entity handling a fixed amount of jobs. A framework exploiting DRL to solve multi-objective problems in large-scale datacenters is discussed in Wang et al. (2019). In this case, the authors manage simultaneous workloads offered to an IaaS while pursuing time completion and cost goals. Even if there are no prior works dealing with DRL for the computation of VM placement strategies in datacenters, the literature includes many works witnessing its flexibility to support the use of cloud computing in many emerging and challenging scenarios. Among the others, we mention the dynamic activation of fog nodes in green radio access networks (Sun et al. 2018), the offload of cloud devices (Chen et al. 2018), and the task distribution in vehicular networks (Zhao et al. 2020).

The use of machine learning is not limited to the computation of proper VM allocations to prevent wastage of resources. For instance, Xu et al. (2012) propose a framework for the autonomic configuration of servers and appliances to find the best trade-off between utilization and SLA levels. A further use of mechanisms based on artificial intelligence is presented by Kumar and Singh (2018), where predictions of the workload offered to the datacenter are carried out. This information is critical for different aspects, ranging from planning of resources and maintenance cycles, to feeding frameworks for computing VM-to-PM mappings. Another example can be found in Yuan et al. (2020), where future predictions of requests are exploited to perform placement in geographically distributed nodes, also by considering multiple levels of computation, i.e., cloud and edge. Moreover, the authors take into account mobility of users by minimizing the impact of network-oriented metrics like the access latency.

Another important tool to optimize the resource utilization of datacenters exploits metaheuristic approaches (see, e.g., Tsai and Rodrigues 2013 for a comprehensive survey on their application to cloud scheduling and Donyagard Vahed et al. 2019 for a review on multi-objective placement mechanisms). The work of Ferdaus et al. (2014) proposes an ant colony optimization metaheuristic to solve VM consolidation problems. We point out that, differently from our work, the authors do not consider placement and they focus on the allocation of RAM, CPU, and network I/O rather than pursuing more general goals.

3 Definition of the VM placement problem

Along the lines of Gaggero and Caviglione (2019), we consider a datacenter running an IaaS made up of M PMs. Each server is assumed to host at most N VMs, modeled as a bundle of CPU, disk, and network resources (Ma et al. 2012; Bobroff et al. 2007; Gaggero and Caviglione 2016). Their utilization is quantized to certain values depending on predefined service plans (Mills et al. 2011). This allows to partition VMs in a number S of classes that differ in the amount of resources available to users (Sugerman et al. 2001; Papadopoulos and Maggio 2015; Gaggero and Caviglione 2016). According to the IaaS paradigm, users request a desired number of VMs with a given lifetime. A VM is said to be "deployed" if it has been successfully created over a PM. As also done by Lango (2014) and by Gaggero and Caviglione (2019), we assume that the datacenter can always satisfy all incoming requests.

As previously pointed out, the goal of the placement procedure is to select the most suitable PMs where to deploy VMs requested by users, in order to optimize one or multiple, often conflicting, objectives. The mapping between PMs and VMs influences performances. If too many VMs are
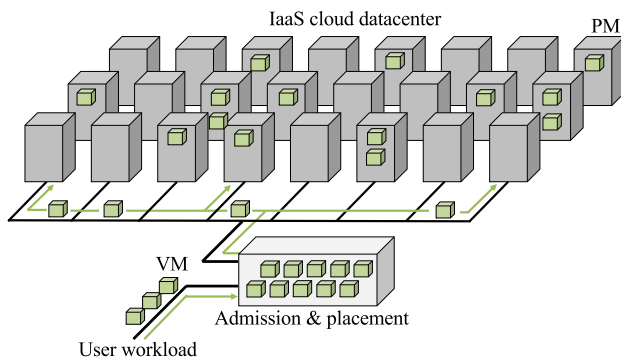
Fig. 1 Reference scenario for the VM placement problem

deployed on the same PM, they may compete one with the others, thus reducing the overall amount of resources made available through virtualization. This is usually referred to as co-location interference. Moreover, PMs could become temporarily unavailable due to hardware or software issues. For instance, a PM may experience local outages or requiring a reboot for software rejuvenation (Pearce et al. 2013; Machida et al. 2012). As a consequence, all the hosted VMs become unavailable as well. An additional feature to take into account is the possible presence of applications requiring to run VMs in a secure environment (Caron and Cornabas 2014; Jhawar et al. 2012). This may prevent the coexistence of VMs running on the same PM.

Figure 1 portraits the reference scenario considered in this paper and the overall system architecture. In more detail, users produce a workload composed of new VM requests, which are collected by the "Admission & placement" module. The latter is in charge of buffering all VM requests at each time step, and it implements a sort of admission control module, i.e., users are not able to directly instantiate VMs or interact with low-level hardware composing the datacenter. Then, this module determines the best placement for requested VMs in PMs of the IaaS cloud datacenter.

We now formalize the considered multi-objective placement problem as a sequence of optimization problems to be solved to satisfy VM requests at different time instants.

3.1 The sequence of optimization problems

We consider a discrete-time representation of the IaaS datacenter, where snapshots of how VMs are mapped over PMs are taken at discrete-time instants $t = 1, \ldots, T$, with T denoting a given time frame. At each time step, we assume to have a number of new VM requests given by $L^t \in \mathbb{N}$. According to the online nature of the VM placement problem, we assume that at time t all VMs requested up to $t-1$ have been already deployed and that the L^t new requests at time t are known. The rth new VM requested by users at time t, where

$r = 1, \ldots, L^t$ and $t = 1, \ldots, T$, is represented through the following input quantities:

- $\hat{y}_r^t \in \{1, \ldots, S\}$ is the class of the VM;
- $\hat{c}_r^t \in [0, 1]$, $\hat{d}_r^t \in [0, 1]$, and $\hat{n}_r^t \in [0, 1]$ are the percentages of requested CPU, disk, and network, respectively;
- $\hat{a}_r^t \in \mathbb{N}$ is the VM lifetime;
- $\hat{s}_r^t \in \{0, 1\}$ denotes the presence or absence of security requirements if equal to 1 or 0, respectively.

As previously pointed out, PMs may suffer from hardware or software issues. To take this into account, we introduce the quantity $g_i^t \in [0, 1]$ for the ith PM, $i = 1, \ldots, M$, $t = 1, \ldots, T$, which represents the probability of having it correctly running. It accounts for both outages and temporary unavailability due to software rejuvenation. Moreover, we consider the heterogeneity of PMs available in the datacenter through the parameters $b_i^c \in (0, 1]$, $b_i^d \in (0, 1]$, and $b_i^n \in (0, 1]$, $i = 1, \ldots, M$, representing the percentages of CPU, disk, and network capacity, respectively, of the ith PM with respect to the most powerful PM in the datacenter, for which the parameters are equal to 1.

To determine the best PM where to deploy new VMs requested by users at each time step $t = 1, \ldots, T$, we apply the placement method described in Sect. 4 according to a sequential approach that consists in placing the L^t new VM requests one at a time. This sequential approach may lead to suboptimal allocations depending on the chosen order of VMs, but it allows a fine granularity since it is possible to select the most suitable placement method for each new VM rather than applying the same criterion for all VMs requested at a certain time step.

From time 1 to T, a sequence of T multi-objective optimization problems have to be solved, and the solution of a given problem influences the following ones. The "connection" among the various problems is established by proper constraints, as discussed below. In more detail, let us denote by Problem-(t) the multi-objective optimization problem that has to be solved at time t to find the best PMs where to deploy all the L^t new VMs requested by users at the same time step. A possible model for Problem-(t) can be found in Gaggero and Caviglione (2019). However, in this paper we introduce a novel formulation for Problem-(t) that considers the sequence of VM assignments at time t. In this new approach, VMs are selected one at a time to determine the most appropriate placement decision. Figure 2 sketches a sequence of optimization problems that have to be solved at three generic time steps t, $t + 1$, and $t + 2$, together with the sequential placement of VM requests within each problem. The order in which VMs are placed is chosen within the optimization problem through suitable decision variables, as detailed below.
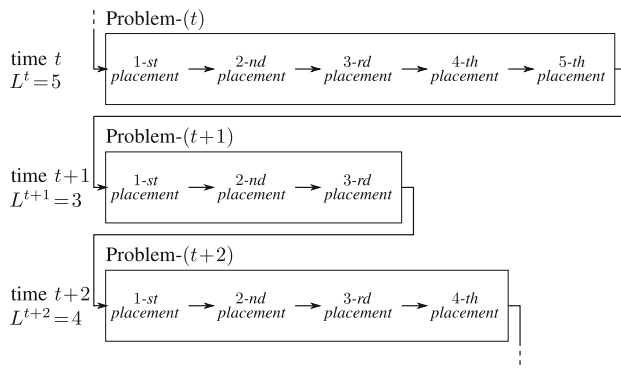
Fig. 2 Example of the sequence of optimization problems to be solved to place new VM requests at the generic time steps t, $t+1$, and $t+2$. The various problems are connected one with the others, and the solution of a given problem influences the solution of the following ones. Within a given problem, VM requests are in turn placed sequentially

Let us now focus on Problem-(t), where $t = 1, \ldots, T$. We introduce the following variables to represent the VM-to-PM mapping, where $i = 1, \ldots, M$, $j = 1, \ldots, N$, and $r = 1, \ldots, L^t$. To account for the aforementioned sequential deployment of VMs, we denote the lth placement within the tth optimization problem, where $l = 1, \ldots, L^t$ and $t = 1, \ldots, T$, with the superscripts l and t.

- $y_{ij}^{l,t} \in \{1, \ldots, S+1\}$ is a discrete variable representing the class of the jth VM on the ith PM. If it is not deployed, $y_{ij}^{l,t} = S+1$, i.e., $S+1$ is a fictitious class denoting a non-deployed VM, introduced to reduce the notation overhead;
- $c_{ij}^{l,t} \in [0, 1]$, $d_{ij}^{l,t} \in [0, 1]$, and $n_{ij}^{l,t} \in [0, 1]$ are real variables denoting the percentages of CPU, disk, and network, respectively, used by the jth VM on the ith PM. For a non-deployed VM, such quantities are equal to 0;
- $s_{ij}^{l,t} \in \{0, 1\}$ is a binary variable denoting the presence or absence of security requirements for the jth VM on the ith PM if equal to 1 or 0, respectively;
- $a_{ij}^{l,t} \in \mathbb{N}_+$ is an integer variable specifying the remaining lifetime of the jth VM on the ith PM. When it is equal to 0, the VM is not deployed or it has completed its lifecycle;
- $u_{ij}^{l,t} \in \{0, 1\}$ is a binary variable equal to 1 if a new VM is deployed at time t as jth VM on the ith PM; otherwise, it is equal to 0;
- $v_r^{l,t} \in \{0, 1\}$ is a binary variable equal to 1 if the rth new VM requested by users at time t is deployed. These variables implicitly determine the order in which requested VMs are placed.

Since the various placement problems are solved sequentially for each $t = 1, \ldots, T$ (see Fig. 2), when solving Problem-(t) all the variables indexed by the superscript $t-1$ are fixed to the values obtained as solution of Problem-$(t-1)$.

Problem-(t), $t = 1, \ldots, T$, reads as follows.

$$\max J_\alpha^t \tag{1}$$

subject to

$$\sum_{j=1}^{N} c_{ij}^{l,t} \le b_i^c, \quad i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \tag{2}$$

$$\sum_{j=1}^{N} d_{ij}^{l,t} \le b_i^d, \quad i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \tag{3}$$

$$\sum_{j=1}^{N} n_{ij}^{l,t} \le b_i^n, \quad i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \tag{4}$$

$$\sum_{j=1}^{N} s_{ij}^{l,t} \le 1, \quad i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \tag{5}$$

$$u_{ij}^{l,t} \ge \chi(a_{ij}^{l-1,t}) - 1, \quad i = 1, \ldots, M,$$
$$j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{6}$$

$$u_{ij}^{l,t} \le 1 - \chi(a_{ij}^{l-1,t}), \quad i = 1, \ldots, M,$$
$$j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{7}$$

$$\sum_{i=1}^{M} \sum_{j=1}^{N} u_{ij}^{l,t} = 1, \quad l = 1, \ldots, L^t, \tag{8}$$

$$v_r^{l,t} - v_r^{l-1,t} \ge 0, \quad r = 1, \ldots, L^t, \quad l = 1, \ldots, L^t, \tag{9}$$

$$\sum_{r=1}^{L^t} \left(v_r^{l,t} - v_r^{l-1,t} \right) = 1, \quad l = 1, \ldots, L^t, \tag{10}$$

$$y_{ij}^{l,t} = y_{ij}^{l-1,t} + u_{ij}^{l,t} \sum_{r=1}^{L^t} \hat{y}_r^t \left(v_r^{l,t} - v_r^{l-1,t} \right),$$
$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{11}$$

$$c_{ij}^{l,t} = c_{ij}^{l-1,t} + u_{ij}^{l,t} \sum_{r=1}^{L^t} \hat{c}_r^t \left(v_r^{l,t} - v_r^{l-1,t} \right),$$
$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{12}$$

$$d_{ij}^{l,t} = d_{ij}^{l-1,t} + u_{ij}^{l,t} \sum_{r=1}^{L^t} \hat{d}_r^t \left(v_r^{l,t} - v_r^{l-1,t} \right),$$
$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{13}$$

$$n_{ij}^{l,t} = n_{ij}^{l-1,t} + u_{ij}^{l,t} \sum_{r=1}^{L^t} \hat{n}_r^t \left(v_r^{l,t} - v_r^{l-1,t} \right),$$
$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{14}$$

$$s_{ij}^{l,t} = s_{ij}^{l-1,t} + u_{ij}^{l,t} \sum_{r=1}^{L^t} \hat{s}_r^t \left(v_r^{l,t} - v_r^{l-1,t} \right),$$
$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \tag{15}$$

$$a_{ij}^{l,t} = a_{ij}^{l-1,t} + u_{ij}^{l,t} \sum_{r=1}^{L^t} \hat{a}_r^t \left(v_r^{l,t} - v_r^{l-1,t} \right),$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad l = 1, \ldots, L^t, \quad (16)$$

where J_α^t is a suitable cost function that will be defined in Sect. 3.2 and the function $\chi(\cdot)$ is such that $\chi(z) = 1$ if $z \neq 0$ and $\chi(z) = 0$ otherwise. The solution of Problem-(t) determines the allocation of all VMs requested by users at the time step t. At time $t + 1$, a new optimization problem, i.e., Problem-($t + 1$), has to be solved to find the best placement of the VM requests at the same instant, and so on up to a given decision horizon.

Let us now describe in detail the various constraints of Problem-(t). Constraints (2), (3), and (4) ensure that the amount of resources (in terms of CPU, disk, and network, respectively) of each PMs is not exceeded. Equation (5) accounts for the need of enforcing security requirements. As said, VMs may need to run in a secured or isolated environment to avoid information leakage or prevent attacks. In other words, (5) guarantees that VMs with security requirements cannot be mixed on the same PM. Equations (6) and (7) guarantee that only one VM of index j exists on the ith PM. They are equality constraints of the kind $u_{ij}^{l,t} = 0$ if $\chi(a_{ij}^{l-1,t}) = 1$, i.e., if another VM with the same pair (i, j) is deployed; otherwise, they are trivially satisfied. Constraint (8) ensures that, for a given l, a new VM requested at time t is placed on one, and only one, PM. Equations (9) and (10) guarantee that only one new VM is chosen to be placed among those that have not yet been deployed for each value of l, where $l = 1, \ldots, L^t$. Lastly, constraints (11)–(16) set the values of the variables $y_{ij}^{l,t}$, $c_{ij}^{l,t}$, $d_{ij}^{l,t}$, $n_{ij}^{l,t}$, $s_{ij}^{l,t}$, and $a_{ij}^{l,t}$ according to the corresponding input quantities of the VM chosen to be placed among those that are not yet deployed at time t depending on the decision variables $u_{ij}^{l,t}$ and $v_r^{l,t}$. The variables $y_{ij}^{l,t}, c_{ij}^{l,t}, d_{ij}^{l,t}, n_{ij}^{l,t}, s_{ij}^{l,t}$, and $a_{ij}^{l,t}$ represent a snapshot of the VM-to-PM mapping after the placement of l VMs at time t, which is related to the value after the placement of $l - 1$ VMs through constraints (11)–(16).

Clearly, if $l = 1$ we have to define $v_r^{0,t}$ in (9), (10) together with $y_{ij}^{0,t}, c_{ij}^{0,t}, d_{ij}^{0,t}, n_{ij}^{0,t}, s_{ij}^{0,t}$, and $a_{ij}^{0,t}$ in (6), (7), (11)–(16). Concerning the former, we let $v_r^{0,t} = 0$ for all $r = 1, \ldots, L^t$, while the latter are equal to the values of the corresponding variables obtained after the placement of all new VMs requested by users at the previous time step, i.e., they are obtained from the solution of Problem-($t - 1$), as follows for $t = 2, \ldots, T$:

$$y_{ij}^{0,t} = \chi \left(a_{ij}^{L^{t-1},t-1} \right) y_{ij}^{L^{t-1},t-1},$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad (17)$$

$$c_{ij}^{0,t} = \chi \left(a_{ij}^{L^{t-1},t-1} \right) c_{ij}^{L^{t-1},t-1},$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad (18)$$

$$d_{ij}^{0,t} = \chi \left(a_{ij}^{L^{t-1},t-1} \right) d_{ij}^{L^{t-1},t-1},$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad (19)$$

$$n_{ij}^{0,t} = \chi \left(a_{ij}^{L^{t-1},t-1} \right) n_{ij}^{L^{t-1},t-1},$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad (20)$$

$$s_{ij}^{0,t} = \chi \left(a_{ij}^{L^{t-1},t-1} \right) s_{ij}^{L^{t-1},t-1},$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N, \quad (21)$$

$$a_{ij}^{0,t} = \max \left(0, a_{ij}^{L^{t-1},t-1} - 1 \right),$$

$$i = 1, \ldots, M, \quad j = 1, \ldots, N. \quad (22)$$

According to the definition of the function $\chi(\cdot)$, the right-hand side of (17)–(21) is equal to 0 if $a_{ij}^{L^{t-1},t-1} = 0$, i.e., if the jth VM on the ith PM has concluded its lifecycle or is not deployed. Equation (22) accounts for the remaining lifetime of VMs, which is decreased of one unit from time $t - 1$ to t.

The values of $y_{ij}^{0,0}, c_{ij}^{0,0}, d_{ij}^{0,0}, n_{ij}^{0,0}, s_{ij}^{0,0}$, and $a_{ij}^{0,0}$ are given initial conditions representing the initial state of the datacenter.

For $t = 1, \ldots, T$, Problem-(t) is a mixed-integer nonlinear one with several unknowns and constraints even for small production-quality datacenters. Therefore, finding a solution is very computationally expensive, especially when complying with real-time requirements. The approach presented in Sect. 4 uses a heuristic method based on DRL to obtain a good approximate solution to the problem.

3.2 Goals of the placement procedure

In this section, we define the cost function J_α^t of Problem-(t), $t = 1, \ldots, T$. Toward this end, the following three competing objectives may be identified for the considered placement procedure: (i) minimize the effects of hardware/software outages, (ii) minimize co-location interference, and (iii) minimize power consumption. A suitable trade-off among the aforementioned goals has to be searched for, as pursuing goals (i) and (ii) may lead to a placement of VMs requiring a large amount of power due to the "spread" of VMs over PMs needed to mitigate the effects of resources becoming unavailable due to malfunctions and co-location interference. On the contrary, the search for an energy-aware placement through goal (iii) may cause too aggressive packings of VMs over PMs, with the consequence of poor fault tolerance properties in case of outages or many VMs aggressively competing and wasting resources due to co-location interference.

The first goal regards the minimization of the effects of hardware/software outages. Usually, they generate the so-called churning behavior of PMs, which causes PMs entering and leaving the pool of available servers due to hardware

or network issues and software rejuvenation. Thus, for the placement procedure, it is convenient to avoid deploying VMs over unreliable PMs (i.e., any server i with a small g_i^t) as much as possible. In fact, if a PM becomes unavailable, all the hosted VMs will become unavailable as well, thus degrading the quality perceived by users or the subscribed SLA. To minimize the effect of churn, the following quantity has to be maximized:

$$\rho_i^{l,t} := \prod_{j=1}^{N} \left(\chi(a_{ij}^{l,t}) \, g_i^t + 1 - \chi(a_{ij}^{l,t}) \right),$$
$$i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \quad t = 1, \ldots, T.$$

Of course, $\rho_i^{l,t} \in [0, 1]$ for all i, l, and t. The rationale is that the more VMs are deployed on the same PM, the higher is the impact of churn. Therefore, $\rho_i^{l,t}$ is low if the number of VMs deployed over an unreliable PM is large. Instead, if g_i^t is close to 1 (i.e., the ith PM is more reliable), also $\rho_i^{l,t}$ approaches 1.

Concerning reduction of co-location interference, it is known that the deployment of too many VMs over the same PM may lead to contentions of the underlying resources, thus reducing overall performances. To take this into account, along the lines of Gaggero and Caviglione (2019) and Caviglione and Gaggero (2021), we define an interference matrix $\Theta \in [0, 1]^{(S+1) \times (S+1)}$, whose elements θ_{hk} measure the interference between the classes h and k of two VMs deployed on the same PM. A value close to 0 models a large interference between the classes h and k, while a number close to 1 denotes a small interference. The overall interference experienced by the ith PM is measured through the following quantity to be maximized:

$$\eta_i^{l,t} := \frac{1}{N} \sum_{j=1}^{N} \prod_{z=1, z \neq j}^{N} \theta_{y_{ij}^{l,t} y_{iz}^{l,t}},$$
$$i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \quad t = 1, \ldots, T.$$

We have $\eta_i^{l,t} \in [0, 1]$, and the larger the interference, the smaller the $\eta_i^{l,t}$.

Concerning minimization of power consumption, we point out that the power required to operate a datacenter is proportional to the number of active PMs, i.e., servers with at least one running VM. In fact, a PM without active VMs can be put in the sleep state to save power. We introduce the following quantity measuring the number of PMs in the datacenter with no running VMs:

$$\omega^{l,t} := M - \sum_{i=1}^{M} \chi \left(\sum_{j=1}^{N} a_{ij}^{l,t} \right),$$
$$l = 1, \ldots, L^t, \quad t = 1, \ldots, T.$$

Large values of $\omega^{l,t}$ indicate a reduced power consumption.

Overall, the cost function J_α^t to be maximized in Problem-(t), accounting for the placement of all VMs requested by users at time t, is the weighted sum of the previously defined quantities, i.e.,

$$J_\alpha^t := \alpha_1 \sum_{i=1}^{M} \rho_i^{L^t,t} + \alpha_2 \sum_{i=1}^{M} \eta_i^{L^t,t} + \alpha_3 \, \omega^{L^t,t},$$
$$t = 1, \ldots, T, \qquad (23)$$

where α_1, α_2, and α_3 are given positive coefficients. The subscript α denotes the parameterization of the cost function with respect to such coefficients.

4 Multi-objective placement using deep reinforcement learning

In this section, we describe the proposed heuristic approach based on DRL to solve the VM placement problem stated in Sect. 3. The goal of this method is the choice of the most appropriate heuristic, among a set of possible alternatives, to place each VM requested by users at the various time steps. In the following, we refer to this approach as "DRL-based VM placement," or DRL-VMP for short.

As pointed out also in Sect. 2, DRL belongs to the family of reinforcement learning methods, which are well suited to dealing with sequential decision making problems like the one introduced in Sect. 3. In more detail, they are based on an agent interacting with an environment in discrete steps through actions taken according to an observation of the state of the environment. As the result of an action, the environment returns a reward, which is a scalar value measuring the effectiveness of the action. The goal is the maximization of the total reward computed as the sum of rewards obtained at each iteration. Unlike supervised learning, rewards do not provide a label for a correct or incorrect answer. Instead, they only consist of a measure of the behavior of the agent, which has to gain insight into the environment through exploration in order to take better actions iteration after iteration. Usually, reinforcement learning approaches model the environment as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, x^0)$ (Szepesvári 2010), where $\mathcal{X}$ is the observation space, $\mathcal{A}$ is the action space, p is the distribution describing the MDP dynamics, and x^0 is the observation of the state at $t = 0$. If the dynamics p is known and $\mathcal{X}$, $\mathcal{A}$ are finite, countable sets with low cardinality, then value iteration or policy iteration methods (Pashenkova et al. 1996) can be used as tabular approaches to solve the MDP, i.e., to find an optimal policy for the agent. However, if one or more of such assumptions do not hold, we have to resort to approximate techniques such as deep Q-learning or policy gradient-based methods (Ivanov
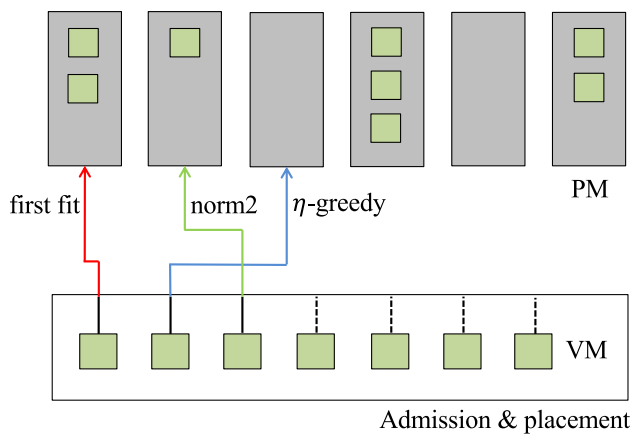
Fig. 3 Toy example illustrating the placement of VMs via the selection of the various heuristics

and D'yakonov 2019), which can deal with large state and action spaces.

In Sect. 4.1, we detail how we modeled the VM placement problem as an MDP by describing our implementation of actions, observations, and rewards. In Sect. 4.2, we describe the algorithm used to find an approximate solution, i.e., the rainbow deep Q-network (DQN) algorithm (Hessel et al. 2018), which employs neural networks to generalize observations and is well suited to dealing with an action space characterized by low cardinality, as in our application.

4.1 Actions, observations, and reward for VM placement

As previously pointed out, the proposed approach based on DRL selects, at each time step and for each VM requested by users, the "best" method to place VMs among a set of available heuristics (the actions), according to a representation (an observation) of the state of the datacenter. The placement problem (the environment) returns a scalar value (the reward) that depends on the performance indices introduced in Sect. 3.2. As detailed in the following, the proposed DRL approach chooses among the available actions in order to maximize the total reward, i.e., to achieve an optimal policy.

In more detail, we focus on six different VM placement heuristics. Three of them are novel and introduced in this paper. They aim at optimizing a single component of the cost function J_α^t in (23) in a greedy way. For this reason, we refer to them as ρ-greedy, η-greedy, and ω-greedy depending on whether they focus on the first, second, and third term in (23), respectively. The other three heuristics are reimplementations (with some minor adaptations to our application) of techniques drawn from the literature on bin packing, i.e., first fit, dot product, and norm2 (Panigrahy et al. 2011).

Figure 3 depicts a toy example illustrating the characteristics of the proposed approach. In more detail, for each

requested VM, the "Admission & placement" block selects the best heuristic looking for a trade-off among the various goals defined in Sect. 3.2. For the sake of simplicity, in the figure we limit the selection among first fit, norm2, and η-greedy, neglecting dot product, ρ-greedy, and ω-greedy. Concerning the first VM request, first fit is selected to save energy: In fact, the VM is placed in the first available PM ignoring co-location interference. The second and third VMs are placed with η-greedy and norm2 heuristics to prevent performance decays and balance power and resource requirements, respectively.

The simplest heuristic is first fit, in which the deployment of a VM is performed on the first available PMs satisfying capacity constraints. Instead, both the dot product and norm2 techniques exploit more elaborated criteria to place VMs. They also consider PMs until a VM placement is possible; however, for the ith PM, dot product sorts VMs that are not yet deployed at time t according to the projection of the used resources over the capacity of the PM. Formally, for all $l = 1, \ldots, L^t$ and $t = 1, \ldots, T$, VMs are sorted in non-ascending order by computing the quantity

$$\hat{c}_r^t \sum_{j=1}^{N} c_{ij}^{l-1,t} + \hat{d}_r^t \sum_{j=1}^{N} d_{ij}^{l-1,t} + \hat{n}_r^t \sum_{j=1}^{N} n_{ij}^{l-1,t} + \hat{s}_r^t \sum_{j=1}^{N} s_{ij}^{l-1,t}$$

for each requested VM r at time t that can be deployed in the lth placement. Then, VMs are considered in the aforementioned sorted order until a placement is performed. The norm2 heuristic is similar to dot product, but sorting of VMs is performed in non-descending order according to the squared norm of the difference between resources required by new VMs and the sum of utilization of PMs. Formally, for the ith PM at given l and t, such norm is computed as

$$\left(\hat{c}_r^t - \sum_{j=1}^{N} c_{ij}^{l-1,t} \right)^2 + \left(\hat{d}_r^t - \sum_{j=1}^{N} d_{ij}^{l-1,t} \right)^2 + \left(\hat{n}_r^t - \sum_{j=1}^{N} n_{ij}^{l-1,t} \right)^2 + \left(\hat{s}_r^t - \sum_{j=1}^{N} s_{ij}^{l-1,t} \right)^2$$

for each requested VM r at time t that can be deployed in the lth placement.

Let us now focus on the ρ-greedy, η-greedy, and ω-greedy heuristics. As said, they aim at minimizing the effect of churn, co-location interference, and power consumption, respectively. Their pseudo-code is reported in Algorithm 1. In more detail, Algorithm 1 refers to a generic heuristic named β-greedy, which represents either ρ-greedy, η-greedy, or ω-greedy depending on the value of the quantity $\beta_i^{r,l,t}$, where $i = 1, \ldots, M, t = 1, \ldots, T, l = 1, \ldots, L^t$, and r is such

that

$$1 \leq r \leq L^t, \tag{24}$$

$$v_r^{l-1,t} = 0, \tag{25}$$

$$\hat{c}_r^t + \sum_{j=1}^{N} c_{ij}^{l-1,t} \leq b_i^c, \tag{26}$$

$$\hat{d}_r^t + \sum_{j=1}^{N} d_{ij}^{l-1,t} \leq b_i^d, \tag{27}$$

$$\hat{n}_r^t + \sum_{j=1}^{N} n_{ij}^{l-1,t} \leq b_i^n, \tag{28}$$

$$\hat{s}_r^t + \sum_{j=1}^{N} s_{ij}^{l-1,t} \leq 1. \tag{29}$$

Equations (24)–(29) are satisfied by any VM r that can be allocated on the ith PM in the lth placement at time t.

In particular, Algorithm 1 corresponds to the ρ-greedy heuristic if

$$\beta_i^{r,l,t} = K_i^{l,t} \, g_i^t \, \rho_i^{l-1,t}. \tag{30}$$

Instead, it corresponds to the η-greedy heuristic if

$$\beta_i^{r,l,t} = K_i^{l,t} \left(\eta_i^{l-1,t} - \frac{1}{N} \left(1 - \prod_{z=1}^{N} \theta_{\hat{y}_r^t y_{iz}^{l-1,t}} \right) \right). \tag{31}$$

Lastly, it corresponds to the ω-greedy heuristic if

$$\beta_i^{r,l,t} = K_i^{l,t} \left(g_i^t \, \rho_i^{l-1,t} \right) \left(\eta_i^{l-1,t} - \frac{1}{N} \left(1 - \prod_{z=1}^{N} \theta_{\hat{y}_r^t y_{iz}^{l-1,t}} \right) \right)$$
$$\left(\min \left(\hat{c}_r^t + \sum_{j=1}^{N} c_{ij}^{l-1,t}, \hat{d}_r^t + \sum_{j=1}^{N} d_{ij}^{l-1,t}, \hat{n}_r^t + \sum_{j=1}^{N} n_{ij}^{l-1,t} \right) \right)^2. \tag{32}$$

The quantities in (30), (31), or (32) are used in Algorithm 1 to evaluate the impact of deploying the rth VM on the ith PM in the lth placement at time t, and they have to be maximized. The term (30) measures the effect of churn resulting from the allocation of the rth VM on the ith PM. Instead, (31) evaluates the updated value of co-location interference after the placement of the rth VM on the ith PM, i.e., a fictitious VM of class $S + 1$ denoting a non-deployed machine is replaced by the rth VM in the computation of the index accounting for co-location interference. Lastly, (32) considers the impact of the deployment of the rth VM on the ith PM by computing the square of the minimum used capacity among CPU, disk, and network of the ith PM, weighted by the factors in (30)

Algorithm 1 β-greedy heuristic

1: **procedure** β- GREEDY(l, t)
2: $\beta_{\max} \leftarrow -\infty$
3: $r^\star, i^\star \leftarrow 0$
4: **for** $i = 1, \ldots, M$ **do**
5:   $\xi_i^{l,t} \leftarrow \{ \hat{y}_{r'}^t : r' \in L^t, \text{ satisfying (24)–(29)} \}$
6:   **for each** $\hat{y} \in \xi_i^{l,t}$ **do**
7:    **let** $r : \hat{y}_r^t = \hat{y}$, satisfying (24)–(29)
8:    **if** $\beta_i^{r,l,t} > \beta_{\max}$ **then**
9:     $\beta_{\max} \leftarrow \beta_i^{r,l,t}$
10:     $r^\star \leftarrow r$
11:     $i^\star \leftarrow i$
12:    **end if**
13:   **end for**
14: **end for**
15: place the $r^\star$th VM at time t on the $i^\star$th PM
16: **end procedure**

---

and (31) in order to avoid too power-efficient deployments, which may degrade user experience.

The quantity $K_i^{l,t}$ in (30), (31), and (32) is defined as follows:

$$K_i^{l,t} := \frac{\omega^{l,t}}{M} \left(b_i^c + b_i^d + b_i^n \right) + 3 \left(1 - \frac{\omega^{l,t}}{M} \right),$$
$$i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \quad t = 1, \ldots, T.$$

This term is the convex combination of $b_i^c + b_i^d + b_i^n$ and 3 (note that $b_i^c + b_i^d + b_i^n \leq 3$ for all $i = 1, \ldots, M$) and acts as a weighting factor in (30), (31), and (32). In particular, if the datacenter is characterized by few deployed VMs, $1 - \frac{\omega^{l,t}}{M}$ approaches 0, and therefore, $K_i^{l,t}$ is close to $b_i^c + b_i^d + b_i^n$. On the contrary, if many VMs are deployed, $1 - \frac{\omega^{l,t}}{M}$ approaches 1; hence, $K_i^{l,t}$ is close to 3. Therefore, the smaller is the number of active PMs in the datacenter, the larger is the influence of the capacities of the ith PM on $\beta_i^{r,l,t}$. As the number of active PMs grows, the capacities of PMs are less relevant for an effective deployment of VMs. This is taken into account by $K_i^{l,t}$, which is progressively less dependent on the capacities of the ith PM. Thus, as $1 - \frac{\omega^{l,t}}{M}$ approaches 1, $\beta_i^{r,l,t}$ is more dependent on the other factors in (30), (31), and (32), which consider the quality of the VM-to-PM mapping, expressed in terms of churn, co-location interference, and power consumption, respectively.

Let us now describe in detail the steps of Algorithm 1. The goal is to select, among the L^t new VMs requested at time t and that can be deployed in the lth placement, the best one and the PM where to deploy it. In other words, we have to select the best pair $(r^\star, i^\star)$, denoting that the lth placement at time t consists in deploying the rth VM request on the ith PM. The algorithm iterates over PMs searching for the pair $(r^\star, i^\star)$ yielding the optimal value $\beta_{i^\star}^{r^\star,l,t}$. In particular, for the ith PM, we consider all VMs requested at time t that can

be deployed in the lth placement on the ith PM, i.e., such that conditions in (24)–(29) are satisfied.

As regards computational complexity of Algorithm 1, a crucial aspect is related to step 5. This statement retrieves the class of VMs since there may be up to $Y_i^{l,t} \leq L^t$, where $Y_i^{l,t} := |\xi_i^{l,t}|$, VM classes for any i, l, t. Afterward, Algorithm 1 iterates over them (steps 6–13) by evaluating the allocation of any requested, not yet deployed VMs of the considered class (step 7). Since the for cycle starting at step 4 is executed M times and the one starting at step 6 is executed $O(Y^{l,t})$ times, where $Y^{l,t} = \max_{i=1,\ldots,M}\{Y_i^{l,t}\}$, Algorithm 1 has a complexity equal to $O(M\,Y^{l,t})$. Note that a further speedup consists in performing the iteration at steps 4–14 over a set initially containing M PMs, and progressively removing from such set any PM that cannot host further VMs requested at time t still to be deployed after the placement of the rth VM at step 15. We denote such PMs as (l,t)-full. An (l,t)-full PM can be identified in $O(1)$ whenever the computation of $\xi_i^{l,t}$ yields $|\xi_i^{l,t}| = 0$. This allows to progressively reduce the number of iterations of the for cycle at steps 4–14. Finally, we point out that Algorithm 1 does not consider the case when a VM placement fails since the datacenter can always satisfy incoming requests, as discussed in Sect. 3.

Concerning state observations adopted in the proposed DRL-VMP approach, they represent a snapshot of the resources used by PMs available in the datacenter for $l = 1, \ldots, L^t$ and $t = 1, \ldots, T$. Since any fixed ordering of PMs does not impact the information contained in the observation, the way data are provided to the DQN (e.g., through a vector or matrix) does not affect the information content. Thus, we aggregate information in a $4M$-dimensional vector, where, for $i = 1, \ldots, M$, elements indexed by $4i - 3$, $4i - 2$, $4i - 1$, and $4i$ are given by $\sum_{j=1}^{N} c_{ij}^{l,t}$, $\sum_{j=1}^{N} d_{ij}^{l,t}$, $\sum_{j=1}^{N} n_{ij}^{l,t}$, and $\sum_{j=1}^{N} s_{ij}^{l,t}$, respectively. In other words, the observation contains aggregate information on VMs deployed on the various PMs, providing a compact representation of the state of the datacenter, as well as a measure of the impact of new deployments. Such information is represented by juxtaposing used CPU, disk, network, and security requirements for each PM in the observation vector. This representation enables to use a feed-forward neural network architecture, disregarding the convolutional layers typical of DQN applications. This unburdens the hyper-parameter tuning and the overall computational cost.

For the lth placement at time t, the reward signal is the following:

$$R^{l,t} := \alpha_1 \sum_{i=1}^{M} \left(\rho_i^{l,t} - \rho_i^{l-1,t}\right) + \alpha_2 \sum_{i=1}^{M} \left(\eta_i^{l,t} - \eta_i^{l-1,t}\right)$$
$$+ \alpha_3 \sum_{i=1}^{M} \zeta_i^{l,t}, \quad l = 1, \ldots, L^t, \quad t = 1, \ldots, T,$$

where the only element different from zero in the sums over i is the one related to the PM chosen for the lth VM placement at time t. The coefficients α_1, α_2, and α_3 are the same of the cost function $J_\alpha^{l,t}$ introduced in Sect. 3.2, and

$$\zeta_i^{l,t} := \begin{cases} -1 \text{ if } \sum_{j=1}^{N} \chi\left(a_{ij}^{l,t}\right) = 1 \\ \quad \text{and } \sum_{j=1}^{N} \chi\left(a_{ij}^{l-1,t}\right) = 0, \\ 0 \text{ otherwise}, \end{cases}$$
$$i = 1, \ldots, M, \quad l = 1, \ldots, L^t, \quad t = 1, \ldots, T. \quad (33)$$

The quantity $\zeta_i^{l,t}$ is equal to -1 if the ith PM was in the sleep state before the lth placement since it was hosting no VMs and a new one is deployed on it, i.e., if the PM needs to be waken up to receive a new VM. Otherwise, if the PM was not in the sleep state, it is equal to 0.

The maximization of the total reward aims at maximizing the decrease of the weighted sum of the performance indices $\rho_i^{l,t}$, $\eta_i^{l,t}$, and $\zeta^{l,t}$, for $i = 1, \ldots, M$. In more detail, let us consider the placement of all the L^t new VM requests at time t. The total reward $\sum_{l=1}^{L^t} R^{l,t}$ obtained at the end of such placement is given by

$$\alpha_1 \sum_{l=1}^{L^t} \sum_{i=1}^{M} (\rho_i^{l,t} - \rho_i^{l-1,t}) + \alpha_2 \sum_{l=1}^{L^t} \sum_{i=1}^{M} (\eta_i^{l,t} - \eta_i^{l-1,t})$$
$$+ \alpha_3 \sum_{l=1}^{L^t} \sum_{i=1}^{M} \zeta_i^{l,t}, \quad t = 1, \ldots, T.$$

The previous expression can be further simplified by cancelling out the terms for l from 1 to $L^t - 1$ in the outer sums, yielding

$$\sum_{l=1}^{L^t} R^{l,t} = \alpha_1 \sum_{i=1}^{M} (\rho_i^{L^t,t} - \rho_i^{0,t}) + \alpha_2 \sum_{i=1}^{M} (\eta_i^{L^t,t} - \eta_i^{0,t})$$
$$+ \alpha_3 (\omega^{L^t,t} - \omega^{0,t}), \quad t = 1, \ldots, T. \quad (34)$$

Equation (34) expresses the differences between the performance indices introduced in Sect. 3.2 computed in $l = L^t$ and $l = 0$ at time t. As regards the factor of α_3 in (34), for a fixed t, the difference between the number of active PMs in $l = 0$ and $l = L^t$ is given by $(M - \omega^{0,t}) - (M - \omega^{L^t,t}) = \omega^{L^t,t} - \omega^{0,t}$. According to (33), $\zeta_i^{l,t} = -1$ if the ith PM is activated in the lth placement at time t. Therefore, since each PM can be activated at most once at t, $\omega^{L^t,t} - \omega^{0,t}$ equals $\sum_{l=1}^{L^t} \sum_{i=1}^{M} \zeta_i^{l,t}$. We conclude that maximization of the total reward entails minimization of the decrease of such indices.

4.2 Rainbow DQN

We focus on a DQN-based algorithm to find an approximate solution of the considered decision making problem. Such algorithms, first introduced in the seminal work of Mnih et al. (2013), are a family of off-policy, model-free reinforcement learning methods based on Q-learning (Kaelbling et al. 1996) that exploit neural networks to approximate the Q-function (Szepesvári 2010).

The original DQN algorithm has been highly improved during the past decade. Among the others, we mention the introduction of the prioritized experience replay buffer (Schaul et al. 2016), i.e., a multi-step learning (Sutton and Barto 2018) and double Q-learning adaptation (van Hasselt 2010; van Hasselt et al. 2016), dueling architectures (Wang et al. 2016), and noisy networks (Fortunato et al. 2018) for exploration. Put briefly, the replay buffer allows the decorrelation of samples used for optimization of the loss function. The insertion of a priority in the buffer allows to sample the transitions that contain more information to learn. Moreover, noisy networks overcome limitations of classical ε-greedy exploration policies. In fact, they allow to act according to the policy learned in the most visited states, while ensuring a high level of exploration in the most unvisited ones. Lastly, double Q-learning reduces the overestimation bias of traditional deep Q-learning, while multi-step learning speeds up the learning process by bootstrapping, i.e., approximating Bellman optimality equation using one or more estimated values.

To have an accurate combination of the aforementioned improvements to the original algorithm, in this paper we focus on the Rainbow DQN (see, e.g., Hessel et al. 2018 for a detailed explanation of the approach).

5 Numerical results

To showcase the effectiveness of the proposed approach, we considered a datacenter composed of $M = 500$ PMs, each one able to host up to $N = 5$ VMs. Six classes of VMs were considered alongside the fictitious one introduced in Sect. 3.2. Each class primarily uses only one type of resource among CPU, disk, and network according to two different usage levels: small and large. The resources used by the fictitious class are set to 0. The considered percentages of resource utilization and the interference matrix Θ are reported in Table 1. This matrix was fixed by taking into account that, in general, the more a resource is used by VMs, the larger is the interference. Moreover, two VMs primarily using disk or network resources suffer from a larger interference if compared to a pair of machines predominantly exploiting CPU. The fictitious class 7 does not interfere with other classes.

Concerning the workload, we focused on four scenarios, named A, B, C, and D, capturing specific traits of real-world infrastructures and taken from Gaggero and Caviglione (2019). In particular, Scenario A relies on synthetic data and accounts for a datacenter characterized by a flow of VM requests with a nearly constant mean over time. At each time step, the number of new VMs requested by users was drawn from uniform distributions in the ranges [0, 12] for class-1 and class-2 VMs, [0, 20] for class-3 and class-4 VMs, and [0, 25] for class-5 and class-6 VMs. Scenario B is still based on synthetic data and presents a peak of VM requests triggered by certain events (e.g., the release of a software update). At each time instant, the number of new VM requested by users was taken from uniform distributions in the ranges [0, 19] for class-1 and class-2 VMs, [0, 22] for class-3 and class-4 VMs, and [0, 24] for class-5 and class-6 VMs, with a peak from $t = 40$ to $t = 120$. Scenario C relies again on synthetic data and is characterized by an increase of VM requests around $t = 50$ and a reduction around $t = 100$. Such a workload emulates, for instance, day/night cycles of the IaaS. At each time instant, the number of new VM requested by users was taken from uniform distributions in the ranges [0, 19] for class-1 and class-2 VMs, [0, 22] for class-3 and class-4 VMs, and [0, 24] for class-5 and class-6 VMs. Lastly, Scenario D is based on the workload of a real datacenter properly scaled to match the size of our simulated environment.

In all scenarios, the amount of VMs characterized by security requirements was taken equal to 15% of the total number of requests, while the time frame T was fixed to 168, i.e., we considered 1 week with sampling time equal to 1 h. Moreover, each VM was assumed to be requested by users for a period randomly chosen according to a uniform distribution in the range [5, 20] hours. Following again Gaggero and Caviglione (2019), we focused on datacenters having different hardware equipment in the various scenarios. In particular, Scenario A is characterized by identical PMs, i.e., $b_i^c = b_i^d = b_i^n = 1.0$ for all $i = 1, \ldots, 500$. Instead, Scenarios B, C, and D are based on PMs with different computing capabilities but identical storage and networking capacities, i.e., we chose $b_i^c = 0.8$ for $i = 1, \ldots, 166$, $b_i^c = 0.9$ for $i = 167, \ldots, 333$, $b_i^c = 1.0$ for $i = 334, \ldots, 500$, and $b_i^d = b_i^n = 1.0$ for $i = 1, \ldots, 500$. Lastly, all scenarios are characterized by the same quantity g_i^t defining the probability that the ith PM is correctly running. Since the literature lacks of a common model to account for all possible causes of hardware or software outages (Gaggero and Caviglione 2019), we randomly extracted the values of g_i^t from a uniform distribution in the range [0.8, 1]. The datacenter was initially considered empty, i.e., we fixed all the quantities $y_{ij}^{0,0}$, $c_{ij}^{0,0}$, $d_{ij}^{0,0}$, $n_{ij}^{0,0}$, $s_{ij}^{0,0}$, and $a_{ij}^{0,0}$ defined in Sect. 3.1 equal to zero.

All the numerical results presented in the following were obtained with a simulator written in Python 3 exploiting

Table 1 Interference matrix and percentage of used resources for the considered VM classes

VM class	1	2	3	4	5	6	7	CPU	Disk	Network
1	0.9	0.8	1.0	1.0	1.0	1.0	1.0	0.23	0.05	0.05
2	0.8	0.7	1.0	1.0	1.0	1.0	1.0	0.46	0.05	0.05
3	1.0	1.0	0.7	0.6	1.0	1.0	1.0	0.05	0.20	0.05
4	1.0	1.0	0.6	0.5	1.0	1.0	1.0	0.05	0.40	0.05
5	1.0	1.0	1.0	1.0	0.7	0.6	1.0	0.05	0.05	0.17
6	1.0	1.0	1.0	1.0	0.6	0.5	1.0	0.05	0.05	0.34
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0	0	0

NumPy and PyTorch libraries, using a PC equipped with a 3.6 GHz Intel i9 CPU with 64 GB of RAM and a GeForce GTX 1060 graphics card with 6 GB of RAM.

5.1 Training of the DRL-VMP approach

The training of the DRL-VMP approach was performed by selecting the following coefficients for the cost function J_α^t in (23): $\alpha_1 = 0.2$, $\alpha_2 = 0.2$, and $\alpha_3 = 1.0$. Such a choice reflects the preference of primarily penalizing power requirements of the datacenter while preserving the goals related to mitigation of hardware/software outages and co-location interference. The duration of the training was almost one full day of computing time.

Synthetic data generated starting from the same probability distributions used for Scenario B were used for the training. In fact, this scenario contains a mixed workload with both "steady" and "peaky" behaviors, and therefore, it can be considered as representative of many different use cases.

The parameters of the training were fixed starting from the results reported by van Hasselt et al. (2019), and a suitable tuning was performed to adapt them to our setting. In more detail, the adopted feed-forward network architecture is based on a fully connected hidden layer with 288 units employing rectifier activation functions. This value was determined with a trial-and-error procedure aimed at finding a compromise between effectiveness and computational efficiency. We chose an overall number of iterations equal to 1,000,000, as we observed that a larger number did not improve the quality of the learned policy. Moreover, the agent performs 100,000 iterations before starting backpropagation training with mini-batches of 32 transitions at each step. The multi-step return length was fixed to 3, and the learning rate of the employed loss function optimization algorithm, i.e., Adam (Kingma and Ba 2015), was set to 0.0001. Lastly, the prioritized replay buffer contained the 100,000 last transitions, while the target network was updated every 8000 steps. We found that smaller periods only allowed to achieve suboptimal policies.
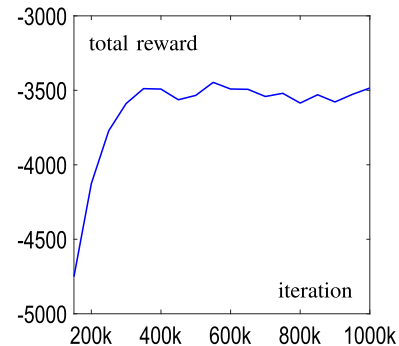

Fig. 4 Total reward obtained during the training of the DRL-VMP approach

Figure 4 shows the total reward for a full training cycle, where the DQN is evaluated every 50,000 steps. The instance used for evaluation was created with the same criteria adopted for generating the training scenarios. The total reward reveals good training properties, as it reaches its maximum after 550,000 iterations, without any significant performance drops throughout the training.

5.2 Performance metrics

In this section, we discuss the main indices adopted to evaluate performance. Such indices, borrowed from Gaggero and Caviglione (2019), are summarized in the following.

The first index quantifies adherence of the IaaS to the subscribed SLA, which depends on how churn of PMs and co-location interference of VMs influence the amount of resource effectively available to users. For the purpose of providing such a measure, we define the following quality of experience (QoE) metric at each time step:

$$
\text{QoE}^t := \frac{\sum_{i=1}^{M} \rho_i^{L^t,t} \sum_{j=1}^{N} \lambda_{ij}^{L^t,t} \prod_{r=1,r\neq j}^{N} \theta_{y_{ij}^{L^t,t} y_{ir}^{L^t,t}}}{\sum_{i=1}^{M} \sum_{j=1}^{N} \lambda_{ij}^{L^t,t}},
$$
$$
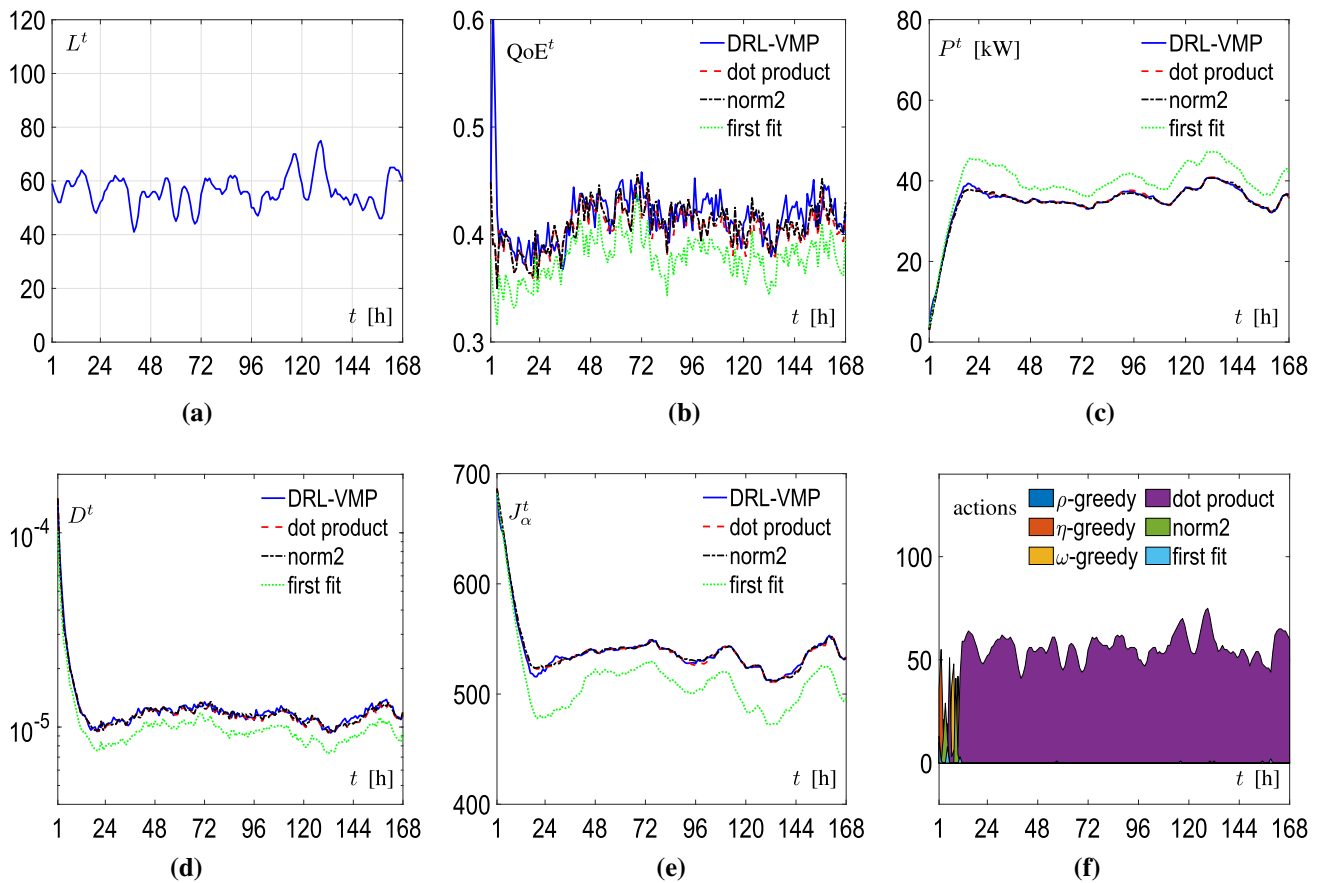t = 1, \ldots, T, \tag{35}
$$

Fig. 5 Simulation results in Scenario A: number of VM requests (**a**), QoE delivered by the datacenter (**b**), consumed power (**c**), density of quality with respect to the required power (**d**), cost function (**e**), and number of VMs placed with a given heuristic by the DRL-VMP approach (**f**)

where $\lambda_{ij}^{L^t,t} := c_{ij}^{L^t,t} + d_{ij}^{L^t,t} + n_{ij}^{L^t,t}$ is the sum of the used CPU, disk, and network capacity of the ith PM at time t after the allocation of all VMs requested at the same step. The numerator of (35) reflects the reduction of resources due to co-location interference and churn, while the denominator is the overall number of requested resources.

The second index measures the power required by the datacenter to operate, which is equal to the sum of the power consumed by PMs. For the sake of simplicity, we assumed that a PM in the sleep state demands a negligible amount of power. Following the reference literature (Gaggero and Caviglione 2016; Kusic et al. 2009; Beloglazov and Buyya 2010), we modeled the power required by a PM as the sum of a constant contribution and a term proportional to the number of deployed VMs, i.e., the following quantity is used:

$$p_i^t := \begin{cases} P_{\max}\left(0.7 + \frac{0.3}{N} n_i^t\right) & \text{if } n_i^t > 0, \\ 0 & \text{otherwise,} \end{cases}$$
$$i = 1, \ldots, M, \quad t = 1, \ldots, T, \tag{36}$$

where $P_{\max}$ is the power required by a PM when N VMs are deployed (this value was chosen equal to 250 W) and $n_i^t :=$

$\sum_{i=1}^{M} \chi\left(a_{ij}^{L^t,t}\right)$ is the number of VMs that are deployed on the ith PM. Thus, the overall power required by the datacenter is given by the following index:

$$P^t := \sum_{i=1}^{M} p_i^t, \quad t = 1, \ldots, T.$$

As previously pointed out, the goals of minimizing the effects of churn and co-location interference (measured by QoE^t) and the overall power consumption (captured by P^t) are conflicting. Thus, we introduce a third index quantifying the trade-off between such goals. More specifically, we consider how a variation of the QoE impacts on the power required by the datacenter and vice versa (see Zhang et al. 2013 for a use case in wireless networks) by introducing the "density" of quality with respect to the required power, as follows:
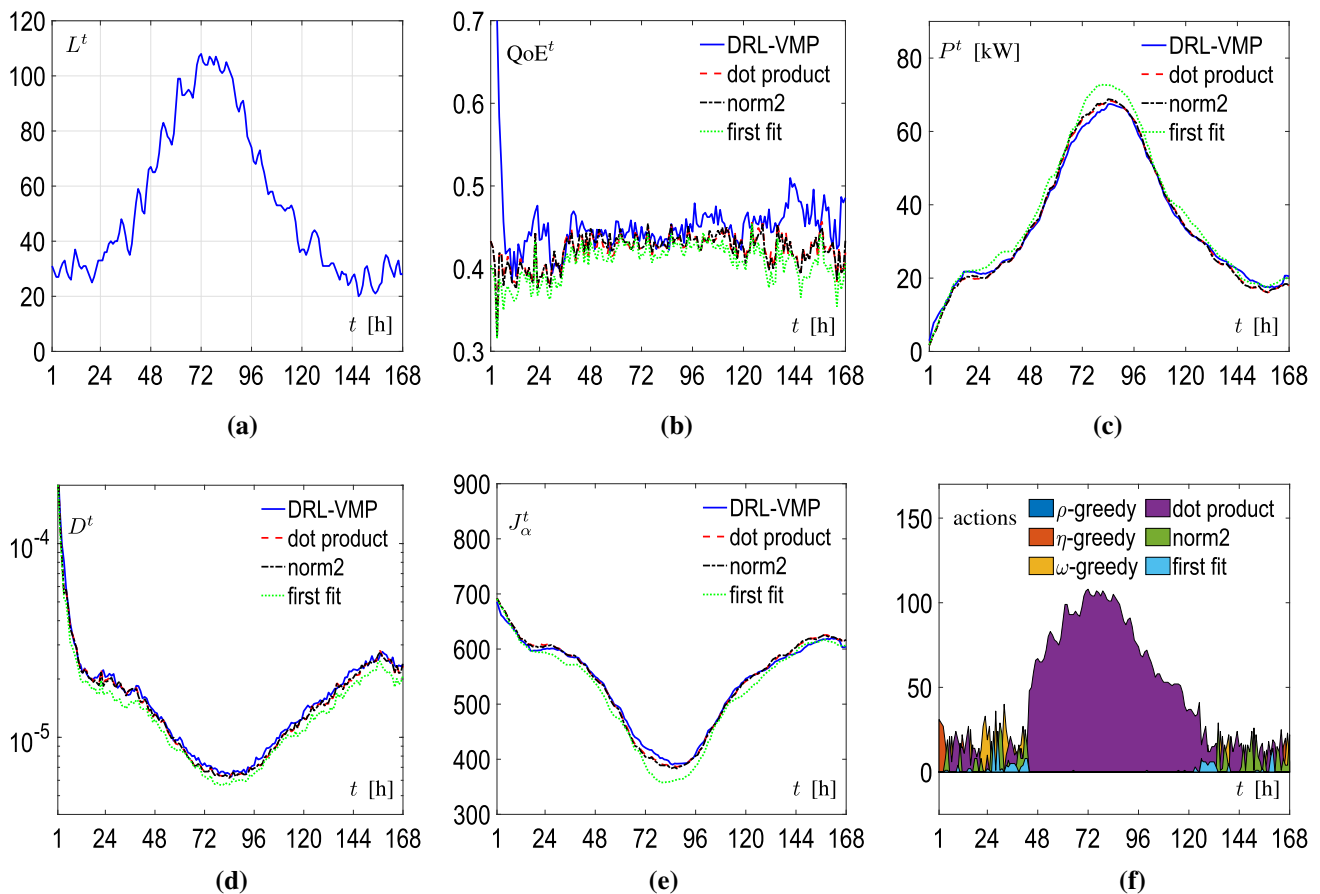
$$D^t := \frac{\text{QoE}^t}{P^t}, \quad t = 1, \ldots, T.$$

Fig. 6 Simulation results in Scenario B: number of VM requests (**a**), QoE delivered by the datacenter (**b**), consumed power (**c**), density of quality with respect to the required power (**d**), cost function (**e**), and number of VMs placed with a given heuristic by the DRL-VMP approach (**f**)

Another index measuring the trade-off among the various goals is given by the cost function J_α^t introduced in Sect. 3.2 for Problem-(t) (see Eq. (23)).

5.3 Simulation results

In this section, we report the results obtained in all the considered scenarios by the proposed DRL-VMP approach in comparison with the first fit, dot product, and norm2 heuristics for VM placement. This analysis allows one to evaluate the behavior of the proposed approach with respect to different workloads.

Figure 5 depicts the results obtained in Scenario A. For all the used heuristics, the performance metrics are influenced by the workload offered by the datacenter. Worst performances characterize first fit, as it simply aims at packing VMs over PMs as much as possible without exploiting "structural" information on PMs at the basis of the cloud service. The other considered heuristics have a similar behavior in terms of consumed power and density. Instead, DRL-VMP guarantees better performance as regards QoE. On the one hand, this could be ascribed to the possibility of switching among

different heuristics to better handle the workload and the current VM-to-PM mapping. On the other hand, its optimized behavior mitigates the impact of bin packing algorithms in terms of too aggressive packing or inefficient allocations as regards utilization of the hardware of PMs. In more detail, it turns out that the most frequently used heuristic by DRL-VMP is dot product. In fact, if we exclude a sort of transient period, DRL-VMP prefers to face fluctuations of the workload by selecting dot product, which is able to account for the various performance goals in a more effective manner owing to its greater sensibility to the multiple criteria used for placement, as detailed in Sect. 4.1.

Different considerations can be done for the case of a peak of VM requests, which characterizes Scenario B. Results are collected in Fig. 6. Specifically, first fit is confirmed as too simple to successfully handle variations in the workload while adhering to multiple, competing performance metrics. Its nature, which is inclined to pack VMs on PMs as much as possible disregarding other objectives, results in higher values for the used power and poorer density (see also the lowest values of the cost when the maximum utilization of the datacenter is reached in the interval from $t = 72$ to $t = 96$). In this
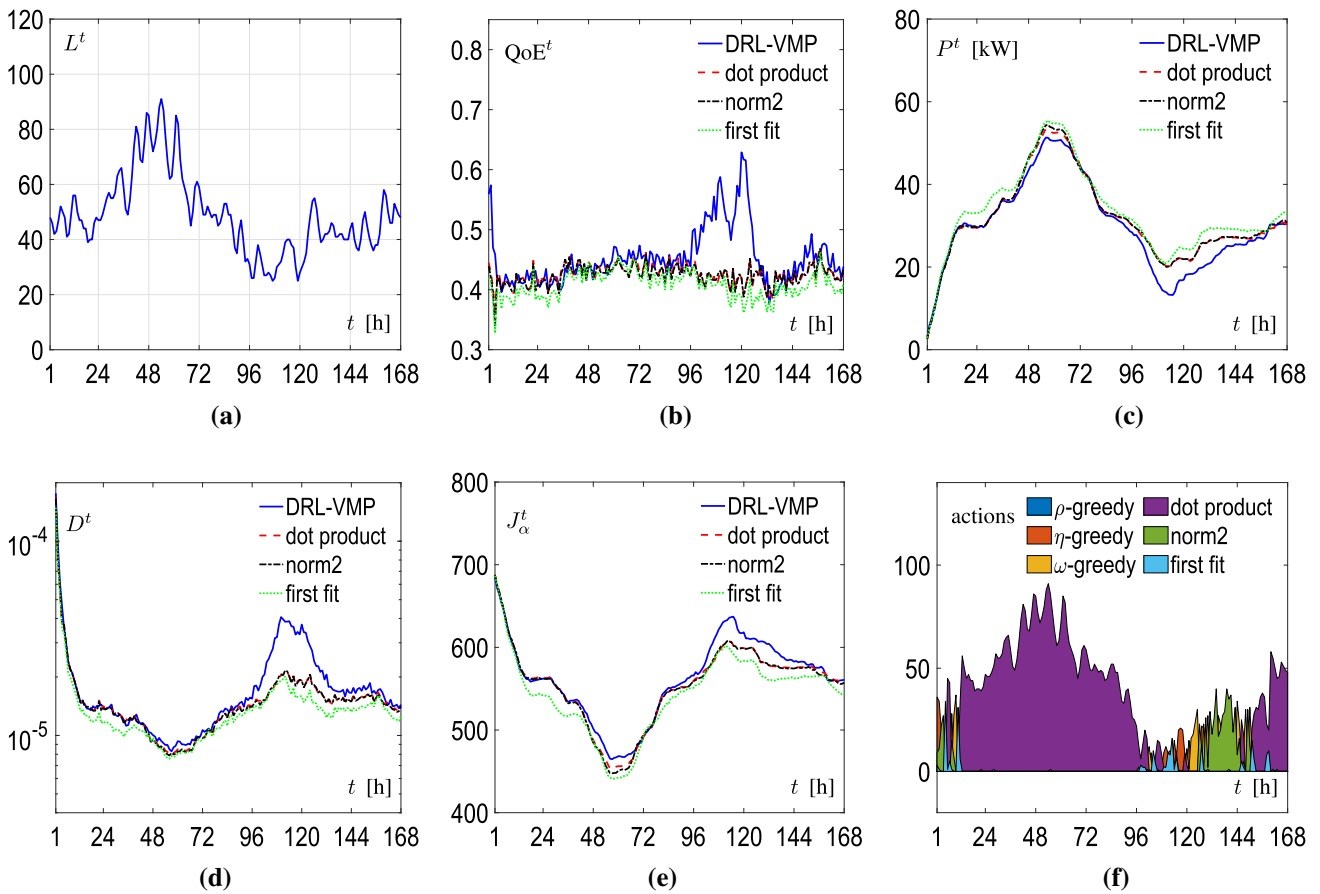
Fig. 7 Simulation results in Scenario C: number of VM requests (**a**), QoE delivered by the datacenter (**b**), consumed power (**c**), density of quality with respect to the required power (**d**), cost function (**e**), and number of VMs placed with a given heuristic by the DRL-VMP approach (**f**)

perspective, DRL-VMP proves to be effective in selecting the most suitable heuristics to face the workload of VMs. First fit is seldom used, and dot product is preferred during the peak since it offers a finer control over the trade-off between QoE and consumed power. This also leads to a minor utilization of the ρ-greedy, η-greedy, and ω-greedy heuristics, as they are specialized to optimize only one metric at a time.

The effectiveness of DRL-VMP is better highlighted in Fig. 7. In this case, the datacenter experiences a peak of requests and then an underutilization epoch. To face such a mixed behavior, DRL-VMP is expected to adapt by changing the heuristics to be used to place the incoming flow of VMs. As shown in the figure, when the workload increases, DRL-VMP privileges again dot product. Instead, when the datacenter is progressively offloaded, DRL-VMP selects the most suitable heuristics to pursue the different performance metrics. A noticeable example of the effectiveness of the approach is the delivered QoE from $t = 96$ to $t = 120$, which is significantly higher if compared to the results provided by the other heuristics. As it can be observed in Fig. 7f, during the aforementioned time interval, the DRL-VMP approach

successfully employs many of the available heuristics at its disposal to achieve these results.

Lastly, Fig. 8 portraits results obtained when the considered datacenter has to serve the real workload of Scenario D. As shown, all heuristics try to place VMs over PMs according to the load of requests with different levels of efficiency. Specifically, first fit confirms its insensitivity to the degree of utilization of PMs, causing poor performance in terms of power and density. Instead, dot product and norm2 can effectively match the workload. However, without a closer inspection of the cost function J_α^t, which is a condensed indicator of the various goals, such results could be misleading. In fact, even if performances of dot product, norm2, and DRL-VMP are similar, the QoE delivered by the cloud datacenter is higher when using machine-learning-enhanced heuristics. In fact, as shown in Fig. 8f, DRL-VMP is capable of decomposing the workload in "slices" (see Fig. 6 for a comparison) and handling them accordingly. Thus, the core of requests is still managed with the dot product placement method, as it is considered able to take into account different goals by exploring the various components of the cost, whereas the beginning and the end of an utilization period are handled by
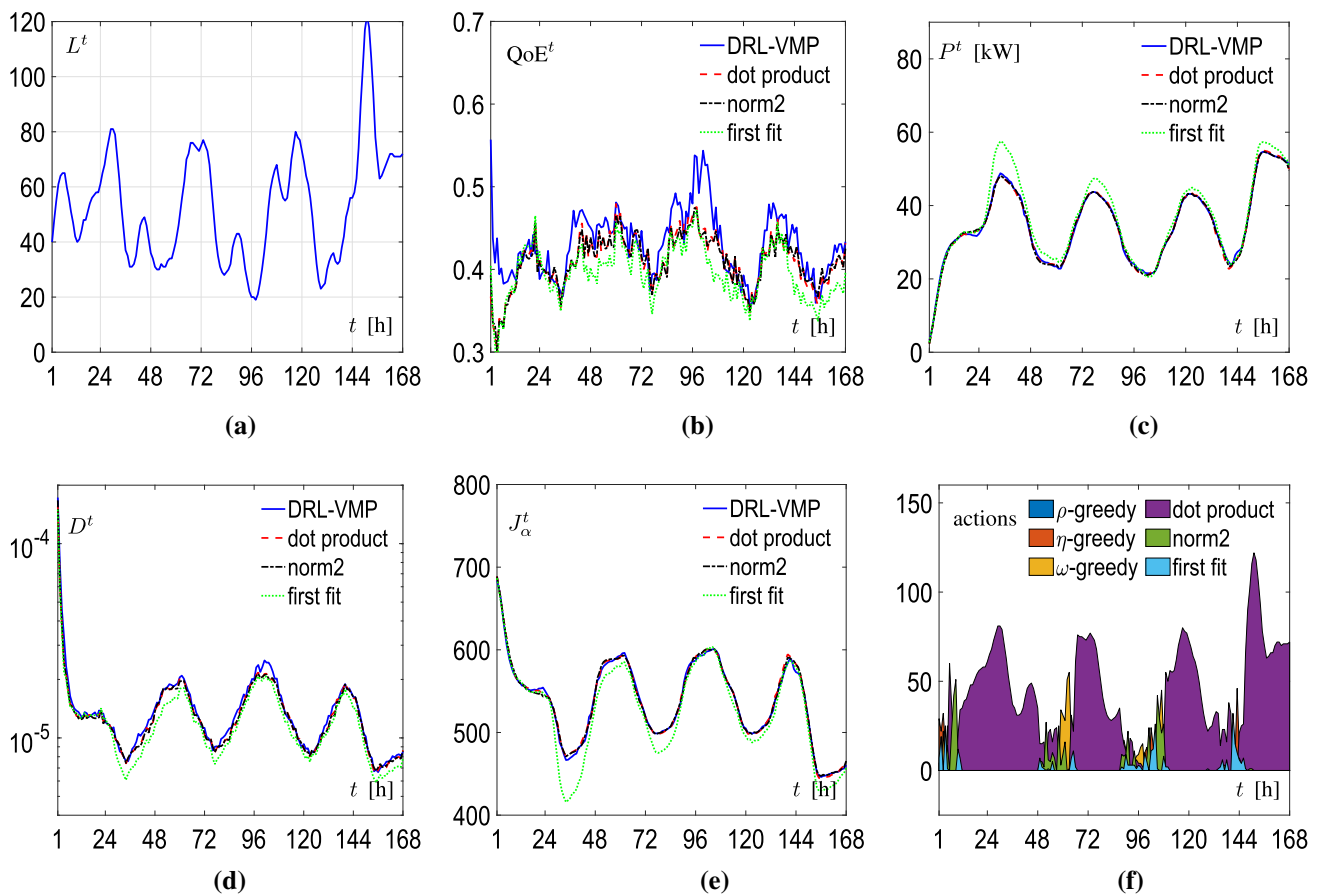
Fig. 8 Simulation results in Scenario D: number of VM requests (**a**), QoE delivered by the datacenter (**b**), consumed power (**c**), density of quality with respect to the required power (**d**), cost function (**e**), and number of VMs placed with a given heuristic by the DRL-VMP approach (**f**)

applying the most suitable placement heuristics to incoming VMs on a case by case basis.

As regards computational requirements, in all the scenarios the time needed to compute the VM-to-PM mapping at a given instant is equal $1.2 \times 10^{-3}, 8.3 \times 10^{-3}$, and 8.3×10^{-3} s on the average for the first fit, dot product, and norm2 heuristics, respectively. The DRL-VMP approach requires about 1.0×10^{-1} s on the average. This amount of time is negligible if compared to the dynamics of an IaaS application, which typically evolves in hours or days. Thus, we conclude that the proposed approach can be effectively used to support managing frameworks and virtualization middlewares commonly used in production-quality datacenters.

5.4 Statistical analysis

To fully evaluate the effectiveness of DRL-VMP in comparison with dot product, norm2, and first fit heuristics for VM placement, we performed the statistical analysis described in the following.

We compared the time series plotted in Figs. 5, 6, 7, and 8 obtained for Scenarios A, B, C, and D, collecting the results

in Tables 2, 3, 4, and 5. These tables report the considered performance metrics, i.e., QoE^t, P^t, D^t, and the cost function J_α^t in (23). The columns "Avg" and "SD" contain the average and standard deviation computed over time, respectively. The columns "AvgDev" denote the percentage deviation of the average of the heuristic in the row with respect to the average of DRL-VMP. The columns p show the p-value returned by the nonparametric Friedman's test (Friedman 1937) that we used to assess the statistical significance of the difference between the average values of DRL-VMP with respect to the one of each competitor heuristic. More specifically, we assume that, if p is smaller than 5%, we can reject the null hypothesis that DRL-VMP and the heuristic generate not significantly different results. We adopted a nonparametric test since we verified that the compared samples are not normally distributed.

Tables 2–5 highlight the best average values for each scenario and metric in bold, as well as the p-values denoting a statistical equivalence between DRL-VMP and the compared heuristics in bold and italic. From Tables 2–5, we can observe the effectiveness of DRL-VMP. In more detail, it is the best heuristic for the QoE^t and D^t metrics in all the scenarios,

Table 2 Results of the statistical analysis for performance evaluation in Scenario A

A

	QoE^t				P^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**0.416**		0.026		34,797.202		5466.138	
dot product	0.405	− 2.62%	0.018	1.13E−08	34,748.095	− 0.14%	5639.489	*0.5807*
norm2	0.409	− 1.78%	0.019	0.0001	**34,678.303**	− 0.34%	5617.250	0.0004
first fit	0.378	− 9.28%	0.019	7.60E−36	39,267.886	12.84%	6857.868	1.08E−36

	D^t				J_α^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**1.340E−05**		1.167E−05		539.458		25.262	
dot product	1.314E−05	− 1.92%	1.204E−05	3.67E−06	539.505	0.01%	26.192	0.0206
norm2	1.326E−05	− 1.06%	1.200E−05	0.0055	**540.026**	0.10%	26.022	0.0136
first fit	1.073E−05	− 19.91%	8.365E−06	2.02E−38	512.076	− 5.07%	33.505	1.08E−36

Table 3 Results of the statistical analysis for performance evaluation in Scenario B

B

	QoE^t				P^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**0.456**		0.055		35,169.047		17,598.026	
dot product	0.422	− 7.42%	0.019	1.10E−17	**35,054.047**	− 0.32%	18,620.489	*0.9368*
norm2	0.422	− 7.41%	0.020	2.08E−16	35,156.131	− 0.03%	18,710.500	*0.8131*
first fit	0.407	− 10.69%	0.022	7.60E−32	37,238.422	5.88%	19,418.526	7.22E−17

	D^t				J_α^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**1.905E−05**		2.5025E−0		537.192		80.289	
dot product	1.811E−05	− 4.93%	2.284E−05	1.30E−09	**537.409**	0.04%	86.121	*0.8774*
norm2	1.805E−05	− 5.26%	2.282E−05	7.92E−13	536.839	− 0.06%	86.569	*0.6434*
first fit	1.608E−05	− 15.58%	1.954E−05	2.02E−34	524.031	− 2.45%	90.932	2.45E−18

Table 4 Results of the statistical analysis for performance evaluation in Scenario C

C

	QoE^t				P^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**0.454**		0.047		**29,814.970**		10,734.515	
dot product	0.425	− 6.32%	0.016	1.13E−04	31,501.815	5.65%	10,132.402	2.31E−08
norm2	0.423	− 6.68%	0.016	3.24E−07	31,661.190	6.19%	10,286.562	7.53E−10
first fit	0.408	− 10.10%	0.022	3.40E−23	33,227.261	11.44%	10,222.868	1.97E−25

	D^t				J_α^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**1.899E−05**		1.616E-05		**561.740**		48.771	
dot product	1.603E−05	− 15.58%	1.395E−05	3.57E−15	554.214	− 1.34%	47.253	1.13E−07
norm2	1.592E−05	− 16.17%	1.379E−05	5.32E−19	553.205	− 1.51%	48.189	1.13E−07
first fit	1.449E−05	− 23.67%	1.255E−05	2.02E−34	543.437	− 3.25%	47.806	3.38E−26

Table 5 Results of the statistical analysis for performance evaluation in Scenario D

D	QoE^t				P^t			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**0.430**		0.040		34,620.654		10,368.102	
dot product	0.410	-4.75%	0.032	4.00E$-$10	**34,494.434**	-0.36%	10,514.905	*0.6219*
norm2	0.409	-4.86%	0.030	1.21E$-$10	34,532.976	-0.25%	10,438.395	*0.0645*
first fit	0.395	-8.08%	0.032	6.25E$-$24	36,462.142	5.31%	11,677.249	4.99E$-$16

	D^t				J^t_α			
	Avg	AvgDev	SD	p	Avg	AvgDev	SD	p
DRL-VMP	**1.496E$-$05**		1.423E$-$05		537.765		48.953	
dot product	1.414E$-$05	-5.45%	1.248E$-$05	1.27E$-$08	**538.428**	0.12%	49.559	*0.6434*
norm2	1.416E$-$05	-5.37%	1.323E$-$05	1.30E$-$09	538.190	0.07%	49.107	*0.0896*
first fit	1.307E$-$05	-12.65%	1.199E$-$05	1.48E$-$28	526.328	-2.12%	56.982	4.84E$-$17

and for all the metrics in Scenario C. We can note from the p-values that the prevalence of DRL-VMP in all these cases is statistically significant. For Scenarios B and D, the best average values for P^t and the cost function J^t_α were obtained by dot product. However, this method is statistically equivalent to DRL-VMP. The only cases in which DRL-VMP is not the best method, or it is not statistically equivalent to the best heuristic, were in Scenario A for P^t and J^t_α, where norm2 obtained the best average results. However, this is not a limitation since the workload in this scenario is characterized by an almost constant mean. In this case, simpler heuristics could be used instead of our approach, which should be deployed when in the presence of aggressively varying loads or flash crowds, such as in Scenarios B, C, and D.

6 Conclusions

In this paper, we have presented a DRL-based approach for the placement of VMs in a cloud datacenter that is able to satisfy different and conflicting performance goals (e.g., the used power and the quality perceived by end users). We have formulated the VM placement problem as a multi-objective, mixed-integer mathematical programming problem, which is very difficult to be solved. Thus, we have developed an approximate approach based on DRL to select, among a set of possible alternatives, the most suitable placement heuristics for each VM, in order to guarantee a trade-off between the performances of the datacenter, its security requirements, and the costs (mainly in terms of used power and required amount of PMs). Results have showcased the effectiveness of the proposed approach, especially when used to handle workloads with major fluctuations. In fact, the agent implementing the selection policy has proved its ability to switch among different heuristics (e.g., those considering a given perfor-

mance metric versus resource-insensitive ones) to guarantee the desired performance criteria.

Future works will be devoted to refining the proposed method, for instance by considering a larger set of heuristics or a policy gradient algorithm. Moreover, another subject of investigation will be the use of Monte Carlo tree search (Browne et al. 2012). With this methodology, optimization could be completely in charge of the agent: the optimization problem could be tackled directly by defining each VM deployment, instead of indirectly acting by means of heuristics. Another prospect of future investigation will aim at considering user mobility. For instance, our approach could be used to place VMs in additional computing layers deployed at the border of the network.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Ahmad RW, Gani A, Hamid SHA, Shiraz M, Yousafzai A, Xia F (2015) A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J Netw Comput Appl 52:11–25

Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) A brief survey of deep reinforcement learning. arXiv:1708.05866

Beloglazov A, Buyya R (2010) Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In: Proceedings of international workshop on middleware for grids, clouds and e-Science, pp 1–4

Bobroff N, Kochut A, Beaty K (2007) Dynamic placement of virtual machines for managing SLA violations. In: International symposium on integrated network management, pp 119–128

Browne C, Powley EJ, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Liebana DP, Samothrakis S, Colton S (2012) A survey of monte carlo tree search methods. IEEE Trans Comput Intell AI Games 4:1–43

Caron E, Cornabas JR (2014) Improving users' isolation in IaaS: Virtual machine placement with security constraints. In: International conference on cloud computing, pp 64–71

Caviglione L, Gaggero M, Cambiaso E, Aiello M (2017) Measuring the energy consumption of cyber security. IEEE Commun Mag 55(7):58–63

Caviglione L, Gaggero M (2021) Multiobjective placement for secure and dependable smart industrial environments. IEEE Trans Ind Inform 17(2):1298–1306

Chen X, Zhang H, Wu C, Mao S, Ji Y, Bennis M (2018) Performance optimization in mobile-edge computing via deep reinforcement learning. In: IEEE vehicular technology conference, pp 1–6

Donyagard Vahed N, Ghobaei-Arani M, Souri A (2019) Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review. Int J Commun Syst 32:1–32

Duan Y, Fu G, Zhou N, Sun X, Narendra NC, Hu B (2015) Everything as a service (XaaS) on the cloud: origins, current and future trends. In: International conference on cloud computing, pp 621–628

Farahnakian F, Liljeberg P, Plosila J (2014) Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning. In: Euromicro international conference on parallel, distributed, and network-based processing, pp 500–507

Farzai S, Shirvani MH, Rabbani M (2020) Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. Sustain Comput Inform Syst, art. no. 100374

Ferdaus MH, Murshed M, Calheiros RN, Buyya R (2014) Virtual machine consolidation in cloud data centers using ACO metaheuristic. In: European conference on parallel processing, pp 306–317

Fortunato M, Azar MG, Piot B, Menick J, Osband I, Graves A, Mnih V, Munos R, Hassabis D, Pietquin O, Blundell C, Legg S (2018) Noisy networks for exploration. In: Proceedings of the international conference on representation learning (ICLR 2018), Vancouver (Canada)

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701

Gaggero M, Caviglione L (2016) Predictive control for energy-aware consolidation in cloud datacenters. IEEE Trans Contr Syst Technol 24(2):461–474

Gaggero M, Caviglione L (2019) Model predictive control for energy-efficient, quality-aware, and secure virtual machine placement. IEEE Trans Autom Sci Eng 16(1):420–432

Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230–1242

Ghobaei-Arani M, Shamsi M, Rahmanian AA (2017) An efficient approach for improving virtual machine placement in cloud computing environment. J Exp Theor Artif Intell 29(6):1149–1171

Ghobaei-Arani M, Rahmanian AA, Shamsi M, Rasouli-Kenari A (2018) A learning-based approach for virtual machine placement in cloud data centers. Int J Commun Syst 31:1–18

Ghobaei-Arani M, Souri A, Baker T, Hussien A (2019) Controcity: an autonomous approach for controlling elasticity using buffer management in cloud computing environment. IEEE Access 7:106912–106924

Guenter B, Jain N, Williams C (2011) Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In: Proceedings of IEEE INFOCOM, pp 1332–1340

Hessel M, Modayil J, van Hasselt H, Schaul T, Ostrovski G, Dabney W, Horgan D, Piot B, Azar M, Silver D (2018) Rainbow: combining improvements in deep reinforcement learning. In: 32nd AAAI conference on artificial intelligence

Ismaeel S, Karim R, Miri A (2018) Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres. J Cloud Comput 7(1):10

Ivanov S, D'yakonov A (2019) Modern deep reinforcement learning algorithms. arXiv:1906.10025

Jhawar R, Piuri V, Samarati P (2012) Supporting security requirements for resource management in cloud computing. In: International conference computational science and engineering, pp 170–177

Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4:237–285

Kaur T, Chana I (2015) Energy efficiency techniques in cloud computing: a survey and taxonomy. ACM Comput Surv 48(2):1–46

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. arXiv:1412.6980

Kumar J, Singh AK (2018) Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Gener Comput Syst 81:41–52

Kusic D, Kephart J, Hanson J, Kandasamy N, Jiang G (2009) Power and performance management of virtualized computing environments via lookahead control. Cluster Comput 12(1):1–15

Lango J (2014) Toward software-defined SLAs. Commun ACM 57(1):54–60

Li Z, Yu X, Yu L, Guo S, Chang V (2020) Energy-efficient and quality-aware VM consolidation method. Future Gener Comput Syst 102:789–809

Liu N, Li Z, Xu J, Xu Z, Lin S, Qiu Q, Tang J, Wang Y (2017) A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In: International conference on distributed computing systems, pp 372–382

Machida F, Xiang J, Tadano K, Maeno Y (2012) Combined server rejuvenation in a virtualized data center. In: International conference on ubiquitous intelligence and computing and international conference on autonomic and trusted computing, pp 486–493

Malekloo M, Kara N (2014) Multi-objective ACO virtual machine placement in cloud computing environments. In: IEEE Globecom workshops, pp 112–116

Ma F, Liu F, Liu Z (2012) Multi-objective optimization for initial virtual machine placement in cloud data center. J Inform Comput Sci 9(16)

Masdari M, Nabavi SS, Ahmadi V (2016) An overview of virtual machine placement schemes in cloud computing. J Netw Comput Appl 66:106–127

Masdari M, Gharehpasha S, Ghobaei-Arani M, Ghasemi V (2019) Bio-inspired virtual machine placement schemes in cloud computing

environment: taxonomy, review, and future research directions. Cluster Comput, 1–31

Mills K, Filliben J, Dabrowski C (2011) Comparing VM-placement algorithms for on-demand clouds. In: Proceedings of international conference on cloud computing technology and Sci, pp 91–98

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller MA (2013) Playing Atari with deep reinforcement learning. arXiv:1312.5602

Panigrahy R, Talwar K, Uyeda L, Wieder U (2011) Heuristics for vector bin packing. Microsoft Research. http://research.microsoft.com/apps/pubs/default.aspx?id=147927

Papadopoulos AV, Maggio M (2015) Virtual machine migration in cloud infrastructures: problem formalization and policies proposal. In: Proceedings of conference on decision and control, pp 6698–6705

Pashenkova E, Rish I, Dechter R (1996) Value iteration and policy iteration algorithms for Markov decision problem. In: AAAI'96: workshop on structural issues in planning and temporal reasoning, Citeseer

Pearce M, Zeadally S, Hunt R (2013) Virtualization: issues, security threats, and solutions. ACM Comput Surv 45(2):1–39

Qin Y, Wang H, Yi S, Li X, Zhai L (2020) Virtual machine placement based on multi-objective reinforcement learning. Appl Intell 50:2370–2383

Schaul T, Quan J, Antonoglou I, Silver D (2016) Prioritized experience replay. arXiv:1511.05952

Shaw R, Howley E, Barrett E (2017) An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers. In: International conference for internet technology and secured transaction, pp 61–66

Song W, Xiao Z, Chen Q, Luo H (2013) Adaptive resource provisioning for the cloud using online bin packing. IEEE Trans Comput 63(11):2647–2660

Srikantaiah S, Kansal A, Zhao F (2008) Energy aware consolidation for cloud computing. In: USENIX HotPower08: Workshop on Power Aware Computing and Systems at OSDI, pp 1–5

Sugerman J, Venkitachalam G, Lim B (2001) Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. In: Proceedings of USENIX annual technical conference, pp 1–14

Sun Y, Peng M, Mao S (2018) Deep reinforcement learning-based mode selection and resource management for green fog radio access networks. IEEE Internet Things J 6(2):1960–1971

Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press, London

Szepesvári C (2010) Algorithms for reinforcement learning. Synth Lect Artif Intell Mach Learn 4(1):1–103

Tsai CW, Rodrigues J (2013) Metaheuristic scheduling for cloud: a survey. IEEE Syst J 8(1):279–291

Usmani Z, Singh S (2016) A survey of virtual machine placement techniques in a cloud data center. Proc Comput Sci 78:491–498

van Hasselt H (2010) Double Q-learning. In: Advances in neural information processing systems, pp 2613–2621

van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double Q-learning. In: 30th AAAI conference on artificial intelligence

van Hasselt H, Hessel M, Aslanides J (2019) When to use parametric models in reinforcement learning? arXiv:1906.05243

Wang Y, Liu H, Zheng W, Xia Y, Li Y, Chen P, Guo K, Xie H (2019) Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. IEEE Access 7:39974–39982

Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, de Freitas N (2016) Dueling network architectures for deep reinforcement learning. arXiv:1511.06581

Xu CZ, Rao J, Bu X (2012) Url: a unified reinforcement learning approach for autonomic cloud management. J Parallel Distrib Comput 72(2):95–105

Xu M, Tian W, Buyya R (2017) A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurr Comput Pract Exp 29(12):e4123

Yuan X, Sun M, Lou W (2020) A dynamic deep-learning-based virtual edge node placement scheme for edge cloud systems in mobile environment. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2020.2974948

Zhang X, Zhang J, Huang Y, Wang W (2013) On the study of fundamental trade-offs between QoE and energy efficiency in wireless networks. Trans Emerg Telecommun Technol 24(3):259–265

Zhang F, Liu G, Fu X, Yahyapour R (2018) A survey on virtual machine migration: challenges, techniques, and open issues. IEEE Commun Surv Tutor 20(2):1206–1243

Zhao J, Kong M, Li Q, Sun X (2020) Contract-based computing resource management via deep reinforcement learning in vehicular fog computing. IEEE Access 8:3319–3329