Journal of Network and Computer Applications 233 (2025) 104049

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Research paper

Joint VM and container consolidation with auto-encoder based contribution extraction of decision criteria in Edge-Cloud environment

Farkhondeh Kiaee [a,*], Ehsan Arianyan [b]

[a] Department of Electrical Engineering, National University of Skills (NUS), Tehran, Iran
[b] ICT Research Institute, Tehran, Iran

ARTICLE INFO

Keywords:
Edge-Cloud environment
Consolidation
Containerization
Energy consumption
Resource management

ABSTRACT

In the recent years, emergence huge Edge-Cloud environments faces great challenges like the ever-increasing energy demand, the extensive Internet of Things (IoT) devices adaptation, and the goals of efficiency and reliability. Containers has become increasingly popular to encapsulate various services and container migration among Edge-Cloud nodes may enable new use cases in various IoT domains. In this study, an efficient joint VM and container consolidation solution is proposed for Edge-Cloud environment. The proposed method uses the Auto-Encoder (AE) and TOPSIS modules for two stages of consolidation subproblems, namely, Joint VM and Container Multi-criteria Migration Decision (AE-TOPSIS-JVCMMD) and Edge-Cloud Power SLA Aware (AE-TOPSIS-ECPSA) for VM placement. The module extracts the contribution of different criteria and computes the scores of all the alternatives. Combining the non-linear contribution learning ability of the AE algorithm and the intelligent ranking of the TOPSIS algorithm, the proposed method successfully avoids the bias of conventional multi-criteria approaches toward alternatives that have good evaluations in two or more dependent criteria. The simulations conducted using the Cloudsim simulator confirm the effectiveness of the proposed policies, demonstrating to 41.5%, 30.13%, 12.9%, 10.3%, 58.2% and 56.1% reductions in energy consumption, SLA violation, response time, running cost, number of VM migrations, and number of container migrations, respectively in comparison with state of the arts.

1. Introduction

In recent years, the rapid advancement of the Internet of Things (IoT) has led to a significant increase in the number of IoT devices and applications. Edge-Cloud computing has gained attention from both academic and industrial communities as an appealing solution for handling the data generated by IoT applications (Ghobaei-Arani et al., 2020; Feng et al., 2022). Edge-Cloud computing enables the timely execution of latency-sensitive tasks, thereby improving user experience.

The Container as a Service (CaaS) offers a new type of service, distinct from traditional models, allowing applications to run in isolated virtual environments while sharing the operating system kernel. Containerization provides advantages such as rapid start-up, high portability, and lightweight properties, making it a suitable solution for managing the complexity of heterogeneous Edge nodes. Containerized resource management frameworks facilitate the efficient allocation of resources and IoT application workloads within a hybrid Edge-Cloud computing environment (Wang et al., 2023a).

Virtualization-based consolidation is a highly effective method for improving energy efficiency in Edge-Cloud environments. By utilizing live migration of VMs and containers, multiple VMs and containers can be grouped onto a minimal set of physical resources, allowing the unused (idle) hosts to be powered down or put into sleep mode (Gholipour et al., 2022). However, heterogeneous resources, dynamic resource demands, and contradictory consolidation goals makes the resource management problem in Edge-Cloud environment a challenging issue. One potential contradictory goal of host consolidation is optimizing resource utilization while minimizing latency. Consolidating hosts aims to use resources more efficiently, but it may lead to increased latency due to sharing resources among multiple VMs. Additionally, striving for energy efficiency might inadvertently result in Service Level Agreement (SLA) violations if the consolidation compromises the performance of critical applications. Balancing these contradictory goals requires careful optimization to ensure an efficient consolidation approach.

As resource demand fluctuates over time, consolidation strategies might necessitate migrating some VMs or containers to a different host to avoid overloading the available hardware resources (Maenhaut et al., 2020). Previously, independent VM migration or container migration was proposed in the literature. The authors in Gholipour et al. (2020)

* Corresponding author.
E-mail addresses: fkiaee@nus.ac.ir (F. Kiaee), ehsan_arianyan@itrc.ac.ir (E. Arianyan).

https://doi.org/10.1016/j.jnca.2024.104049
Received 4 February 2023; Received in revised form 11 September 2024; Accepted 28 October 2024
Available online 5 November 2024

Table 1
Comparison of relevant studies across various parameters.

Work	Method	Edge/ Cloud	VM	Container	Joint VM & Container	Scalability	Latent space	Optimization goals				
								Energy	SLA-aware	Cost	Response time	Co-Location
Beloglazov and Buyya (2012)	PABFD	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗
Horri et al. (2014)	UMC	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗
Arianyan et al. (2015)	TPSA	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓
Pham and Huh (2016)	Cost/ Makespan balance	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗
Piraghaj et al. (2017)	CORHS	✓	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗
Fu et al. (2021)	Nautilus	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✓
Ma et al. (2020)	LBJC	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗
Chen et al. (2018)	First/Best fit	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓
Li and Hu (2019)	DDQN	✗	-✓	✗	✗	✗	✓	✓	✗	✗	✗	✗
Basu et al. (2019)	Qlearning	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗
Mao et al. (2016)	DRL	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗
Tuli et al. (2020)	A3C-R2N2	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗
Wu et al. (2016)	MBFD	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
Jiang and Chen (2018)	SARA	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
Lebre et al. (2018)	VMPlaceS	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
Khan et al. (2020)	EPC	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗
Zakarya et al. (2022)	CoLocateMe	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗
Gholipour et al. (2020)	JVCMMD	✗	✓	✓	✓	✗	✓	✓	✓	✗	✗	✓
This work	AE-TOPSIS-JVCMMD	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

verified that joint VM and container consolidation (when containers are running inside VMs) is more energy-efficient than consolidating of VMs or containers, individually.

The standard VM consolidation problem in cloud data-centers is divided into four main phases (Beloglazov and Buyya, 2012), namely, overloaded host determination, underloaded host determination, VM selection from over-utilized hosts for migration, and new placement determination for migrating VMs. An effective resource allocation strategy can reduce the number of required hosts, decrease energy costs, and minimize SLA violations. Recent studies have increasingly utilized the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) as a multi-criteria decision-making algorithm to address one or more phases of the consolidation problem (Arianyan et al., 2015; Gholipour et al., 2020). However, most TOPSIS-based approaches often assume that the criteria being evaluated are independent, which may not hold true in Edge-Cloud environments. As a result, the outcomes could be skewed in favor of options that perform well on two or more interdependent criteria (Pelegrina et al., 2019).

This paper proposes a novel joint VM and container consolidation solution designed for Edge-Cloud environment. Moreover, the Auto-Encoder (AE) learning approach is utilized to extract the contribution of the initial set of criteria and decorrelate them considering both linear and nonlinear relationships of the decision criteria instead of considering just linear relationships. Then using the reduced decision matrix, the TOPSIS multi-criteria decision approach is used to rank and select the best candidates. The main contributions of this paper are:

(1) Proposing a joint container and VM consolidation solution applicable to the complex Edge-Cloud environment.
(2) Proposing two multi-criteria decision making policies for migrating VMs/containers selection and placement based on TOPSIS method that simultaneously optimizes energy consumption, response time, cost, and SLA violations with minimum number of VMs/containers migrations
(3) Learning a set of independent latent criteria from the observed data using Auto-Encoder (AE) method, which can be used as an alternative latent representation of the original decision matrix for TOPSIS method.
(4) Evaluating the impact of various consolidation approaches that migrate VMs, containers or VMs plus containers (containers running inside VMs) in Edge-Cloud environment on energy efficiency, cost, and performance metrics.

This paper begins by reviewing related works in Section 2. Section 3 introduces the proposed system model. Section 4 presents the proposed solution for joint VM and container resource management in Edge-Cloud environment. Section 5 evaluates the effectiveness of our proposed solutions using the Cloudsim simulator. Finally, Section 6 presents concluding remarks and suggests directions for future research.

2. Related works

There is a wide area of research related to this work from different aspects including Edge-Cloud computing, containerization, and consolidation problems. Table 1 summarizes the comparison of relevant studies with our work across various parameters.

In the Edge-Cloud landscape, the requests from IoT devices can be served by several nodes. Each request could be divided into a set of tasks. The resource management problem determines an optimal assignment of different tasks submitted to be executed on the Edge/Cloud nodes in order to meet QoS requirements (Ghobaei-Arani et al., 2020). Edge-Cloud resource management then might be formulated to a multi-objective optimization (Ma et al., 2020; Tuli et al., 2020; Pham and Huh, 2016; Fu et al., 2021) with minimization of energy, costs, response-time, SLA violation and co-location as shown in Table 1. Co-located VMs on the same host can face significant performance drops, especially when they contend for the same resources (Fu et al., 2021).

Resource management approaches might be classified according to the domain subject to the optimization techniques into two general groups that include heuristic and Machine Learning (ML) methods. In ML-based resource management methods mostly a neural network with many layers have deployed in learning high level features in a latent space from large input spaces as indicated in 8th column of Table 1.

Heuristic methods have been extensively researched in the context of cloud resource management and empirically demonstrated its scalability for large number of tasks and hosts (Ma et al., 2020; Pham and Huh, 2016; Wang et al., 2023b; Li et al., 2020; Fu et al., 2021). The authors in Ma et al. (2020) have considered consolidation problem in Edge-Cloud environment and designed a Load Balancing Joint Migration Cost (LBJC) minimization approach for container placement that minimizes the latency impact of container migration while balancing the load of hosts. A two-stage heuristic resource management

approach for the Edge-Cloud computing paradigm has been proposed in Pham and Huh (2016). The first stage focuses on determining task priority, while the second involves selecting the optimal node based on Earliest Start Time and Earliest Finish Time criteria. The authors in Zakarya et al. (2022) have presented energy, performance, cost, and co-location efficient VM placement and consolidation strategies for running multiple co-located workloads (CoLocateMe). The authors in Lebre et al. (2018) have also examined different VM placement and consolidation techniques by VMPlaceS as a dedicated simulation framework using three types of schedulers: centralized, hierarchical and distributed. The Modified Best Fit Decreasing (MBFD) method have presented in Wu et al. (2016) for VM consolidation, taking into account energy consumption and migration costs, specially performance loss due to downtime. Authors in Jiang and Chen (2018) have proposed the Self Adaptive Resource Allocation (SARA) algorithm which dynamically allocates resources to VMs in an energy efficient manner.

Recent works have explored different ML techniques for dynamic optimization of resource management systems (Soni and Kumar, 2022). In a reinforcement learning (RL) setup, Q-tables or neural networks are built from the actual measurements, and used to approximate Q action-value function (Please refer to Garí et al. (2021) for more information on RL-based methods). Deep learning based methods like Deep Q Learning (DQN) (Li and Hu, 2019; Basu et al., 2019) or other Deep Reinforcement Learning (DRL) establishments which use a neural network with many layers have deployed in learning high level features from large input spaces (Mao et al., 2016; Tuli et al., 2020). A resource manager (Nautilus system) has been developed in Fu et al. (2021) that determines the optimal resource allocation based on RL to capture the contention behaviors in order to minimize co-location, SLA violation and resources usage. A scheduler for Edge-Cloud environments has been presented in Tuli et al. (2020) using Asynchronous-Advantage-Actor-Critic learning based on Residual Recurrent Neural Network (A3C-R2N2). The method considers all tasks and hosts to make scheduling decisions and continuously adapt to the dynamics of the system. However, the developed methods that train an end-to-end network with fixed output layer structure, face problem of limited scalability or can schedule for a fixed number of edge nodes and tasks.

In this paper, the proposed method combines heuristic and ML techniques to take advantage of scalability and automated feature learning. The Auto-Encoder (AE) neural network structure is used to learn the contribution of input criteria in the latent feature space and a heuristic-based method is proposed to address joint VM and container consolidation problem in large scale Edge-Cloud environment.

As containers become more widely employed in cloud environments, it is important to consider containerized workloads. In cloud computing research, various studies have explored the impact of consolidation policies on energy consumption, performance, and costs for workloads running in different environments: VMs (Beloglazov and Buyya, 2012; Arianyan et al., 2015), containers (Piraghaj et al., 2017; Ma et al., 2020; Zakarya et al., 2022), or containers within VMs (Gholipour et al., 2020; Khan et al., 2020). Three distinct consolidation strategies can be evaluated: (i) VM-based – migrating only VMs; (ii) container-based – migrating only containers; and (iii) joint VM and container-based – migrating both containers and VMs. Each strategy (i or ii) has the authority to independently decide on migrating VMs and containers. However, in the context of (iii), a particular workload is hosted on containers running on virtualised resources (inside VMs), and a joint VM and container consolidation policy is used to migrate VMs and containers in combination. Previous researches have primarily focused on VM consolidation and container consolidation, separately and joint VM and container consolidation has rarely explored (Gholipour et al., 2020; Khan et al., 2020).

The authors in Beloglazov and Buyya (2012) have introduced the Power Aware Best Fit Decreasing (PABFD) algorithm for VM placement. This algorithm has been designed to sort VMs in descending order based on CPU utilization demand and assigns each VM to the host that results in the smallest increase in power consumption. Overloaded physical machine detection and VM selection are dynamically carried out using Local Regression (LR) and Minimum Migration Time (MMT) heuristics, respectively.

In Arianyan et al. (2015), the authors have explored the VM consolidation problem, proposing multi-criteria algorithms for the two main stages of standard VM consolidation (Beloglazov and Buyya, 2012): identifying under-loaded hosts and VM placement. Specifically, they have introduced the TOPSIS Power and SLA Aware Allocation (TPSA) policy for VM placement. They have simulated the approach using the CloudSim simulator to evaluate its effectiveness in reducing energy consumption, VM migrations, and SLA violations. This method has been later extended to address the joint consolidation of VMs and containers (Gholipour et al., 2020).

The authors in Piraghaj et al. (2017) have tackled the consolidation problem by modeling the CaaS environment and addressing power optimization through the container CloudSim simulator. They identified three key subproblems for consolidation: detecting over-loaded and under-loaded hosts, selecting containers for migration, and determining container placement. Their experimental results show that the approach identifies over-loaded and under-loaded hosts using thresholds of 70% and 80%, respectively combined with the Maximum Usage (MU) algorithm for selecting the largest container to migrate and their developed correlation host selection (CORHS) algorithm, performs better than other algorithms.

A standard joint VM and container consolidation procedure has been developed in Gholipour et al. (2020) that divides the cloud resource management problem into seven sub-problems including (1,2) detecting over-loaded/under-loaded hosts, (3) deciding whether migrate VMs or containers by identifying which VMs should be migrated from these hosts, (4) selecting VMs from the identified candidate VMs for migration, (5) placing VMs on appropriate destination hosts, (6) selecting containers for migration, and (7) placing containers on appropriate destinations. The authors presented a policy named Joint VM and Container Multi Criteria Migration Decision (JVCMMD) technique for the third sub-problem. Moreover, in order to avoid co-location of VMs that compete for similar resources on a specific host, the VMs with the most correlation with other VMs in the same host were selected for migration in order to decrease the probability of the host to become overloaded by decreasing resource contention due to co-location.

Another joint VM and container consolidation approach has been developed in Khan et al. (2020) that divides the cloud resource management problem into six sub-problems including (1,2) over-loaded/under-loaded host detection, (4,5) containers/VMs selection for migration, (6,7) containers/VM placement. Moreover, this study presents a policy named Energy and Performance Efficient consolidation (EPC) technique for the 4th/5th sub-problem. The developed approach lacks the third stage in comparison to Gholipour et al. (2020) that is a particular stage to control candidate VMs/containers for migration and avoid repeated migrations of VMs/containers.

However, the consolidation algorithms presented in Gholipour et al. (2020) and Khan et al. (2020) are quite basic and fail to consider the most efficient migration strategies concerning energy consumption and performance degradation within a heterogeneous Edge-Cloud environment. Moreover, their investigations is limited to certain correlated decision criteria. These works lack the non-linear contribution extraction of the criteria which motivate us to perform this study.

3. Target system model

The target system model includes three important parts: the Edge-Cloud infrastructure, users with various workloads, and the proposed consolidation framework. An overall view of the system model is depicted in Fig. 1. The consolidation part itself consists of the ‘local manager’, which is executed on all nodes and the ‘global manager’, which is executed on a central host. The numbers in the circles in Fig. 1

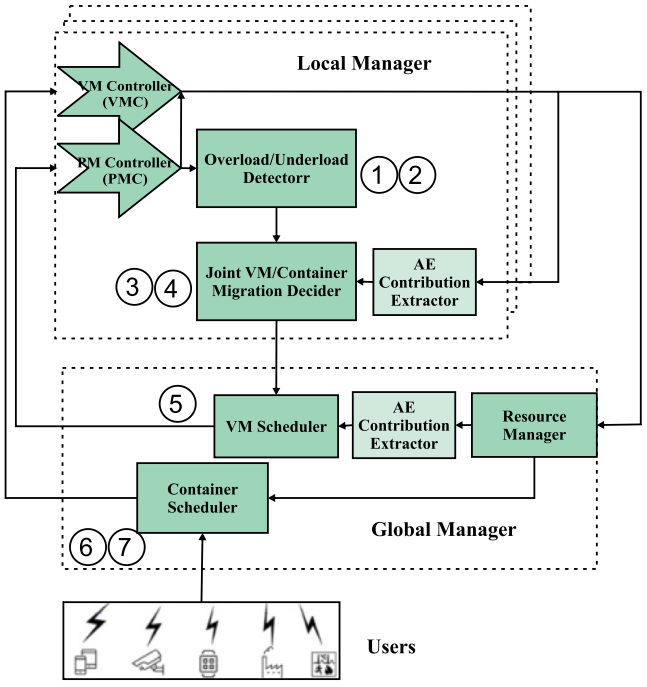

Fig. 1. An overview of the target system model.

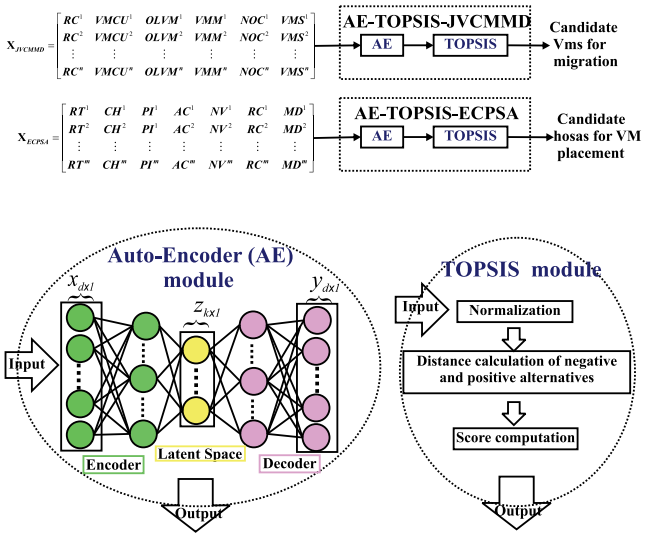

Fig. 2. A schematic diagram of Auto-Encoder (AE) and TOPSIS modules (bottom). Both proposed AE-TOPSIS-JVCMMD and AE-TOPSIS-ECPSA policies take advantage of AE and TOPSIS modules (top). A properly trained AE correctly reconstructs the input while reducing the latent layer dimensionality. The Mean Square Error (MSE) of this reconstruction $l = MSE(x, y) = \Sigma_{i=1}^{d}(x_i - y_i)^2$ provides training loss function. In our work the output of the latent space is used as TOPSIS input criteria.

correspond to the 7 stages of the proposed consolidation approach in Section 4.

Edge is a generic paradigm, which has many candidate architectures. The spectrum of Edge architectures ranges from Fog Computing, Mobile Edge Computing (MEC), and Edge-Cloud to specialized hardware like Field-Programmable Gate Arrays (FPGAs) (Xu et al., 2020) and smartphones (Mateos et al., 2022). The targeted architecture depends on specific use cases and requirements. For our scenarios requiring resource management closer to end-users, Edge-Cloud might be suitable. The system's infrastructure consists of both resource-rich and resource-limited nodes that vary significantly in parameters such as compute power measured in Millions Instructions Per Second (MIPS) and response times. These nodes support a variety of users running different applications, which involve multiple heterogeneous VMs and containers, leading to a dynamic and diverse workload on each host. Edge devices which are closer to the users, offer much lower response times but are constrained by limited computation capacity. Conversely, cloud resources are farther from the users, resulting in higher response times, but they are abundant in resources enhanced computational capabilities, enabling execution of multiple tasks simultaneously.

The local manager exists in each active host and is consisted of five main components that are introduced as follows:

- Virtual Machine Controller (VMC)/Physical Machine Controller (PMC):he VMC is a daemon within each VM that gathers resource information, such as the number of running containers and resource utilization. Similarly, the PMC, located in each physical machine (PM), collects data on resource usage, capacity, and the number and types of VMs running on the PM.
- Overload/Under-load detector: The component determines the under-loaded/overloaded hosts using double threshold's policy — which uses a lower (20%) and an upper (80%) threshold values. This information is sent to Joint VM/Container Multi-criteria Migration Decider component in the Global Manager.
- AE Contribution Extractor: Auto-Encoder (AE) can basically be considered as a non-linear Principle Component Analysis (PCA) using neural networks with auto-associative architecture. It involve three main parts, namely encoder, latent layer, and decoder, whose encoding and decoding weights reach optimal values in the Mean Square Error (MSE) sense. The architecture of the AE Contribution Extractor module is depicted in Fig. 2. The encoder compresses the input criteria into a compact latent layer. The decoder brings the compressed latent section back into the form of the original criteria. The output is then expected to be a proper reconstruction of the input. Latent layer is a key component of the AE and provides data compression to the input with powerful feature extraction capabilities. The decoder section can then be discarded, leaving the latent layer and its encoding weights already able to approach non-linear PCA of input data. The output of latent layer, $\mathbf{Z}_{JVCMMD}$ is used as the input criteria of TOPSIS module of the proposed AE-TOPSIS-JVCMMD approach in Section 4.1. The input criteria of AE, $\mathbf{X}_{JVCMMD}$, (listed in Table 2) are provided by the VMC and PMC. The encoder function Q_{JVCMMD} maps the original input data to the latent output, $\mathbf{Z}_{JVCMMD} = Q_{JVCMMD}(\mathbf{X}_{JVCMMD})$.
- Joint VM/Container Multi-criteria Migration Decider (JVCMMD): The component determines the candidate VMs for migration using the proposed AE-TOPSIS-JVCMMD approach in Section 4.1. It enhances resource utilization by evenly spreading VMs across hosts and reducing the number of active PMs. Migration priorities are determined based on the latent layer criteria of AE Contribution Extractor i.e. $\mathbf{Z}_{JVCMMD}$, allowing VM instances to be migrated from overloaded hosts. Moreover, VM instances can be migrated from a lightly loaded host so those hosts can be turned off, minimizing the total number of active hosts. When candidate VMs for migration are found, the final migrating VMs among candidate VMs are determined using MMT algorithm. JVCMMD immediately informs the migration decisions to the VM Scheduler when there is any VM migration.

The global manager operates on a distinct node that determines the appropriate destinations for the selected containers and VMs to be migrated. It consists of the following components:

- Resource Manager: The resource manager monitors information on both PMs and VMs. It sends the collected decision criteria (according to Table 3) to AE Contribution Extractor to extract

Table 2
Considered criteria in the proposed AE-TOPSIS-JVCMMD policy.

No.	Notation	Criteria	Description	Benefit/Cost
1	RC	Resource correlation	Correlation between a VM's resources and those of other VMs on the same host	Benefit
2	VMCU	VM CPU utilization	The CPU utilization of a VM	Benefit
3	OLVM	Over-Load VM	The total MIPS of containers hosted within a VM	Benefit
4	VMM	VM MIPS	The computing capacity of a VM in MIPS	Cost
5	NOC	Number of container	The number of containers available within a VM	Benefit
6	VMS	VM storage	The storage capacity of a VM	Cost

latent space criteria as the input of the TOPSIS sub-module of VM Scheduler. Similarly, Resource Manager sends the collected information to Container Scheduler.

- AE Contribution Extractor: The general performance of this component is similar to one in local manager. The output of latent layer, $\mathbf{Z}_{ECPSA}$ is used as the input criteria of TOPSIS module of the proposed AE-TOPSIS-ECPSA approach in Section 4.2. The input criteria of AE, $\mathbf{X}_{ECPSA}$, (listed in Table 3) are provided by Physical Machine Controller (PMC). The encoder function Q_{ECPSA} maps the original input data to the latent output, $\mathbf{Z}_{ECPSA} = Q_{ECPSA}(\mathbf{X}_{ECPSA})$.
- VM Scheduler: The VM Scheduler determines the placements of VMs on the servers. The placement decisions are based on the information available from the Resource Manager. The VM Scheduler applies the proposed AE-TOPSIS-ECPSA in Section 4.2 to find a placement for each VM on a server. When the VM Scheduler identifies a suitable placement, it notifies the PMC to start the VM and update the resource data. If a VM cannot be placed appropriately, It sends the details to the Container Scheduler.
- Container Scheduler: The Container Scheduler is responsible for deciding where to place containers on VMs. In the proposed system, every task operates within a container. The CS communicates with the Resource Manager to gather information about the available resources of each VM and applies the correlation host selection (CORHS) policy for container placement. If the containers correlation at the destination host is unavailable, the least full host selection (LFHS) policy serves as a fallback option. When the CS identifies a suitable placement, it informs the VMC to start the container. If no existing VM has sufficient resources to host the container, a new VM needs to be instantiated. To do this, the CS requests the VMC to instantiate a new VM. Using data from the PMC, the VMC determines a placement for the new VM, instantiates the new VM, and updates the Resource Manager. The VMC then launches the container on the newly instantiated VM. Fig. 3 shows the sequence diagram for the proposed architecture.

The proposed method is the resource manager of a specific Edge-Cloud environment which allocates VMs and containers to available hosts based on the predefined goals. It also decides when and which VMs or containers should be migrated from hosts. The details of the proposed method is presented in Section 4.

4. Proposed joint VM and container consolidation for Edge-Cloud environment

The primary limitation of most consolidation solutions in edge-cloud environments is their lack of consideration for a combined container and VM migration policy. To address this shortcoming, this paper introduces a new approach for joint VM and container consolidation, which divides the edge-cloud resource management process into seven sub-problems (corresponding to the numbers in Fig. 1):

(1) Over-loaded host detection: The over-loaded hosts are determined using Local Regression (LR) policy (Beloglazov and Buyya, 2012).
(2) Under-loaded host detection: The under-loaded hosts are determined using Simple Method (SM) policy (Beloglazov and Buyya, 2012).
(3) Candidate VMs/containers determination for migration: The candidate VMs for migration from over-loaded and under-loaded hosts are identified using our proposed TOPSIS based Joint VM and Container Multi-criteria Migration Decision (AE-TOPSIS-JVCMMD) policy. The proposed AE-TOPSIS-JVCMMD takes advantage of AE module to extract the contribution of its input criteria (Section 4.1).
(4) VM selection for migration: Among the candidate VMs, those selected for migration are chosen based on the Minimum Migration Time (MMT) policy (Beloglazov and Buyya, 2012).
(5) VM placement: New destination hosts for selected migrating VMs are determined using our proposed Edge-Cloud Power SLA Aware (AE-TOPSIS-ECPSA) policy that uses AE and TOPSIS modules to extracts the contribution of input criteria and computes the scores of all the host, respectively (Section 4.2). The VMs that were selected for migration are eliminated from the initial candidate VM list and the list of remaining VMs that no suitable destination host is found for them is built.
(6) Container selection for migration: The VMs that still require migration but were not initially selected are handled, and the containers hosted on these VMs are selected for migration using the Maximum Usage (MU) policy (Piraghaj et al., 2017).
(7) Container placement: The new destinations for the chosen containers are determined using the Correlation Threshold Host Selection Algorithm (CORHS), with the Least Full Host Selection Algorithm (LFHS) serving as a fallback option (Piraghaj et al., 2017).

More specifically, in this paper, two novel AE-TOPSIS-JVCMMD policy and AE-TOPSIS-ECPSA policy are proposed for the third and fourth sub-problems, respectively. Our proposed consolidation procedure for migration decision is shown in Algorithm 1. In the next subsections we describe the AE-TOPSIS-JVCMMD and AE-TOPSIS-ECPSA policies in detail.

4.1. AE-TOPSIS based Joint VM and Container Multi-criteria Migration Decision (AE-TOPSIS-JVCMMD) policy

AE-TOPSIS-JVCMMD policy leverages the AE and TOPSIS modules as part of a multi-criteria algorithm, incorporating six criteria outlined in Table 2 for its decision-making process. In other words, AE-TOPSIS-JVCMMD is a multi-criteria decision-making method that selects solutions from a limited set of options by minimizing the distance to the ideal positive point while maximizing the distance from the ideal negative point. The criteria used in this approach can be either benefit-based or cost-based. A higher value for benefit-type criteria or a lower value for cost-type criteria brings the solution closer to the optimal point. The negative of cost type criteria are hence considered as the equivalent benefit one in the proposed AE-TOPSIS-JVCMMD policy.

AE-TOPSIS-JVCMMD computes the score of VMs so that the following conditions exist in the answer:

(1) Resource correlation between the selected VM for migration and other VMs in the same host should be the most: It increases the probability of the host to become overloaded by increasing the correlation between applications that utilize the same resources

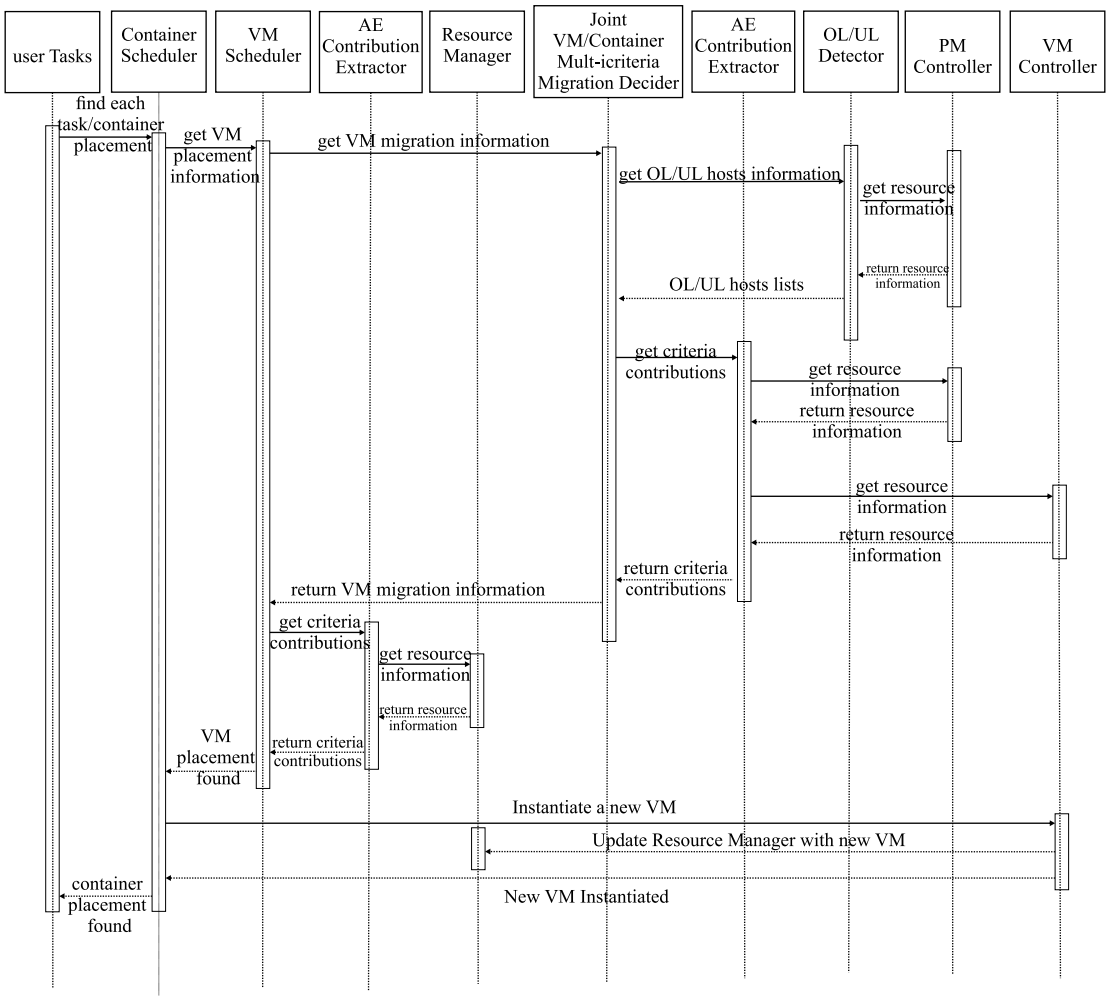

Fig. 3. Sequence diagram for the proposed consolidation approach.

Table 3
Considered criteria in the proposed AE-TOPSIS-ECPSA policy.

No.	Notation	Criteria	Description	Benefit/Cost
1	RT	Response time	Response time of Edge/Cloud nodes hosting VMs	Cost
2	CH	Cost of Hosts	Deployment cost of Edge/Cloud hosts	Cost
3	PI	Power Increase	The increase in power consumption when placing VMs on a host	Cost
4	AC	Available Capacity	Remaining resource capacity of a host	Benefit
5	NV	Number of VMs	Number of VMs running on a host	Cost
6	RC	Resource correlation	Resource correlation between a VM and other VMs on the destination host	Cost
7	MD	Migration delay	The delay caused by migrating VMs to different hosts	Cost

on an oversubscribed host. In order to calculate correlation coefficients, for each pair of VMs (including the VM in question and each other VM on the same host), Pearson correlation coefficient is calculated for the resource usage of the VMs and the average correlation of the target VM with all other VMs on the host is then considered as a decision criterion.

(2) CPU utilization of the selected VM should be the highest: Prioritizing the migration of a VM with higher utilization reduces the likelihood of resource shortages, and consequently, minimizes the risk of violation.

(3) The selected VM is the most over-loaded one due to high resource usage of the containers it hosts: A VM is considered over-loaded if the average computing capacity of the host, calculated per processing elements of both the host and the VM is lower than the total computing capacity utilized by all containers in MIPS. Migrating the over-loaded VM helps minimize the risk of resource shortages on its host and potential SLA violation.

(4) The VM chosen for migration should have the lowest computing capacity (MIPS) possible: Opting for a VM with higher computing power increases the risk of running out of resources for hosting containers.

(5) The selected VM should have the highest number of containers: Choosing such a VM for migration reduces the likelihood of resource contention, which in turn minimizes SLA violations. In other words, VMs with the most containers are prioritized for migration in order to avoid the co-location of VMs that compete for more resources on a specific host (decreasing resource contention due to co-location).

(6) The chosen VM should have the minimum storage capacity possible : By the selecting a VM with the lowest storage capacity (VMS), the migration overhead caused by storage transmission is reduced.

These criteria are used to construct multi-criteria decision matrix,

Algorithm 1 Proposed joint VM and containers consolidation for Edge-Cloud environment

Input: List of all hosts, VMs, and containers
Output: Generate a migration map
Step 1: Identify over-utilized hosts using **LR** policy.
Step 2: Identify under-utilized hosts using **SM** policy.
for each over-utilized/under-utilized host **do**
Step 3: Find candidate VMs for migration using the proposed **AE-TOPSIS-JVCMMD** policy:
Make criteria matrix (Eq. (1)) for running VMs .
Extract contribution matrix using AE (Initialization: Eq. (2)), Online updating: Eq. (3)).
Calculate score of VMs using TOPSIS (Eq. (4)).
Candidate high ranked VMs for migration.
Step 4: Select VMs to migrate from the candidate list, using **MMT** policy.
if VMs are selected **then**
Step 5: Identify appropriate hosts for migrating VMs using proposed **AE-TOPSIS-ECPSA** policy:
Make criteria matrix of Edges/Clouds (Eq. (5)).
Extract contribution matrix using AE (Initialization: Eq. (6), Online updating: Eq. (7)).
Calculate score of hosts using TOPSIS (Eq. (8)).
Place VMs on the top-ranked hosts and include them in the migration map.
Step 6: Identify containers of migrating VMs for which no suitable destination is found for them, using **MU** policy.
Step 7: Place containers using **CORHS/LFHS** policy and include them in the migration map.
end if
end for
return Migration map

$$\mathbf{X}_{JVCMMD} = \begin{bmatrix} RC^1 & VMCU^1 & OLVM^1 & VMM^1 & NOC^1 & VMS^1 \\ RC^2 & VMCU^2 & OLVM^2 & VMM^2 & NOC^2 & VMS^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ RC^n & VMCU^n & OLVM^n & VMM^n & NOC^n & VMS^n \end{bmatrix} \tag{1}$$

where, n is the number of instantiated VMs. Before using $\mathbf{X}_{JVCMMD}$ as input of TOPSIS approach, it is feed-forwarded to the AE network as shown in Fig. 2. In practice, the AE network is trained by two sequential steps of initialization and online updating of the parameters. In the initialization step, the PCA method is applied to the matrix to extract contributions (Seuret et al., 2017). Given a $n \times d$ input criteria matrix ($d = 6$), some criteria are likely to be correlated, since they are taken from the same VM. PCA leverages an orthogonal transformation to a new coordinate system, to get a relatively small number of uncorrelated criteria, which are the principal components. The first principle component indicates the direction of the maximum variance and each subsequent principle component points to the direction of the highest remaining variance, ensuring the orthogonality to all previous components.

Using PCA for automated cloud management has been widely studied in the context of anomaly detection, e.g., Du and Li (2017) and Manimurugan (2021). To the best of our knowledge, there is no prior work in adapting PCA-initialized AE for consolidation method over Edge-Cloud environments.

In order to build the PCA model, eigenvalue decomposition is performed on the covariance matrix, and a set of eigenvectors $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}2, \dots, \mathbf{v}_d)$ is obtained, ordered by their eigenvalues. These eigenvectors define the new axes of the transformed coordinate system. The first principal axis, v_1, aligns with the direction of the greatest variance, while each subsequence principal axes points in the direction of the next highest variance that is orthogonal to the preceding axes. The corresponding eigenvalues are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$.

The principal subspace is the space formed by the first k principal axes in $\mathbf{V}$, denoted as $\mathbf{P}_1 = [\mathbf{v}_1, \dots, \mathbf{v}_k]$, while the residual subspace is the space spanned by the remaining axes, $\mathbf{P}_2 = [\mathbf{v}_{k+1}, \dots, \mathbf{v}_d]$. The cumulative percent variance (CPV) method is used to determine the value of k due to its simplicity and reliable performance (Li et al., 2000). Specifically, for the first principal components, $CPV(l) = \frac{\Sigma_{i=1}^{l} \lambda_i}{\Sigma_{i=1}^{d} \lambda_i} 100\%$, the value of k is selected as: $k = argmin_l(CPV(l) > 90\%)$.

In summary, the proposed AE-TOPSIS-JVCMMD policy consists of three steps:

1. Constructing multi-criteria decision matrix $\mathbf{X}_{JVCMMD}$ from all the criteria assigned to the candidates in time slot t as shown in Eq. (1),
2. Constructing latent space criteria matrix $\mathbf{Z}_{JVCMMD}$ by training AE:
   - Initialization: performing PCA by $\mathbf{X}_{JVCMMD}$ and determining the first k principal components to make weighted matrix
   $$\mathbf{Z}_{JVCMMD} = \mathbf{X}_{JVCMMD}\mathbf{P}_1, \tag{2}$$
   where the weights are eigen vectors associated with principal components $\mathbf{P}_1 = [\mathbf{v}_1, \dots, \mathbf{v}_k]$.
   - After initialization: feed-forwarding $\mathbf{X}_{JVCMMD}$ to the AE network
   $$\mathbf{Z}_{JVCMMD} = Q_{JVCMMD}(\mathbf{X}_{JVCMMD}). \tag{3}$$
3. Performing TOPSIS to acquire score for the ith candidate by the Eq. (4),
   $$S_{JVCMMD} = \frac{\sqrt{\Sigma_{j=1}^{k}(z_j^i - z_j^-)^2}}{\sqrt{\Sigma_{j=1}^{k}(z_j^i - z_j^+)^2} + \sqrt{\Sigma_{j=1}^{k}(z_j^i - z_j^-)^2}}. \tag{4}$$
   where z_i^j represents an entry in the latent space matrix $\mathbf{Z}_{JVCMMD}$, after being normalized by dividing each element by the maximum value in its respective column. Moreover, z_j^+ is the ideal positive solution and z_j^- is the ideal negative solution. The ideal positive solution corresponds to the maximum value of z_j^i, while the ideal negative solution corresponds to the minimum value of z_j^i ($j = 1, 2, \dots, k$ and $i = 1, 2, \dots, n$).

The intuition to use PCA-initialized AE as an contribution extractor of each criterion, is that the target positive and negative ideals of TOPSIS method are most likely calculated with some bias due to the correlation between each dimension in the original space. Thus by transforming the data matrix onto a new latent space, the original coordinates are rotated in a way that the new positive and negative ideals would be ideal solutions in latent space.

For instance, when the dimensionality $d = 2$, Fig. 4 illustrates this scenario. In the left panel of Fig. 4, the CPU utilization (VMCU criterion from Table 2) and storage (VMS criterion from Table 2) for a randomly selected node and its trace from the Bitbrain dataset (Shen et al., 2015) are shown to be strongly correlated. When the TOPSIS method is applied to the latent space, the positive and negative ideal targets are depicted in the right panel of Fig. 4. However, both ideal alternatives illustrated in Fig. 4 (left panel) when the TOPSIS approach is applied in the original observed data. It is clear that applying TOPSIS to the original data does not produce the desired positive and negative ideals. As a consequence, the distance measures are incorrectly calculated, leading to a ranking of alternatives that differs from the intended one. In other words, the original correlated space introduces bias into the determination of the positive and negative ideals.

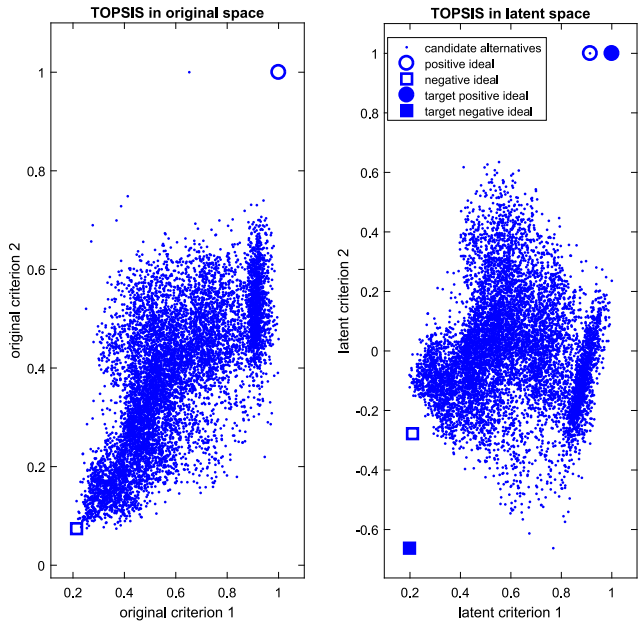

Fig. 4. Graphical intuition behind latent space conversion of TOPSIS criteria in 2-dimensional space.

For instance, when the dimensionality $d = 2$, Fig. 4 illustrates this scenario. In the left panel of Fig. 4, the CPU utilization (VMCU criterion from Table 2) and storage (VMS criterion from Table 2) for a randomly selected node and its trace from the Bitbrain dataset (Shen et al., 2015) are shown to be strongly correlated. When the TOPSIS method is applied to the latent space, the positive and negative ideal targets are depicted in the right panel of Fig. 4.

It is important to note that after initialization, the parameters in the AE are iteratively updated in an online manner with the recently released data using the back-propagate (BP) algorithm. The online updating strategy allows the method to get aware of the latest Edge-Cloud condition such as workload variability and resource availability at the edge and cloud levels and revise its parameters accordingly.

4.2. AE-TOPSIS based Edge-Cloud Power SLA Aware (AE-TOPSIS-ECPSA) policy

AE-TOPSIS-ECPSA policy leverages both AE and TOPSIS modules as part of a multi-criteria algorithm, that takes into account seven criteria outlined in Table 3 during its decision-making process. using the TOPSIS module, The policy evaluates the scores of all potential hosts for VM placements and selects the host with the highest score. The criteria used in AE-TOPSIS-ECPSA policy can be classified as either cost-based or benefit-based. The cost type criteria are hence converted to the equivalent benefit type with negative sign in the proposed AE-TOPSIS-ECPSA method. AE-TOPSIS-ECPSA assigns the highest rank to a host that simultaneously satisfies the following conditions:

(1) The selected host should have the lowest response time: This prioritizes choosing resource-limited edge nodes that are closer to the user for handling medium to lightweight tasks.
(2) The deployment cost should be minimized for the selected host: This favors selecting resource-rich cloud nodes with lower computing cost for complex and resource-intensive tasks.
(3) The power consumption of allocating a VM should be minimized for the selected host.
(4) The selected host should have the maximum available resources: This increases the likelihood of allocating the necessary resources for VMs, thereby reducing the SLA violation rate.
(5) The selected host should have the fewest number of VMs: This minimizes competition for shared resources among VMs, which in turn reduces SLA violations.
(6) The resource correlation between the VM to be allocated and the existing VMs on the selected host should be minimized: This decreases the chance of the host becoming overloaded by reducing the correlation between applications that use the same resources on an oversubscribed host.
(7) The migration delay of the VM to the selected host should be minimized: This reduces SLA violations during the migration process and enables smarter decisions that decreases the number of VM migrations and avoid those with long delays.

The proposed AE-TOPSIS-ECPSA policy consists of three steps:

1. Constructing multi-criteria decision matrix $\mathbf{X}_{ECPSA}$ from all the criteria assigned to the candidate hosts in time slot t,

$$\mathbf{X}_{ECPSA} = \begin{bmatrix} RT^1 & CH^1 & PI^1 & AC^1 & NV^1 & RC^1 & MD^1 \\ RT^2 & CH^2 & PI^2 & AC^2 & NV^2 & RC^2 & MD^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ RT^m & CH^m & PI^m & AC^m & NV^m & RC^m & MD^m \end{bmatrix}, \tag{5}$$

   where, m is the number of edge and cloud hosts. Before using $\mathbf{X}_{ECPSA}$ as input of TOPSIS approach, it is feed-forwarded to the AE network as shown in Fig. 2. In practice, the AE network is trained by two sequential steps of initialization and online updating of the parameters.
2. Constructing latent space criteria matrix by training AE:
   - Initialization: performing PCA by $\mathbf{X}_{ECPSA}$ and determining the first k principal components to make weighted matrix
   $$\mathbf{Z}_{ECPSA} = \boldsymbol{X}_{ECPSA}\boldsymbol{\rho}_1, \tag{6}$$
   where the weights are eigen vectors associated with principal components $\boldsymbol{\rho}_1 = [\upsilon_1, \ldots, \upsilon_k]$.
   - After initialization: feed-forwarding $\mathbf{X}_{ECPSA}$ to the AE network
   $$\mathbf{Z}_{ECPSA} = Q_{ECPSA}(\mathbf{X}_{ECPSA}). \tag{7}$$
3. Performing TOPSIS to acquire score for the ith candidate by the Eq. (8),

$$S_{ECPSA} = \frac{\sqrt{\Sigma_{j=1}^{k}(\zeta_j^i - \zeta_j^-)^2}}{\sqrt{\Sigma_{j=1}^{k}(\zeta_j^i - \zeta_j^+)^2} + \sqrt{\Sigma_{j=1}^{k}(\zeta_j^i - \zeta_j^-)^2}}. \tag{8}$$

   where ζ_i^j represents an entry in the latent space matrix $\mathbf{Z}_{ECPSA}$, after being normalized by dividing each element by the maximum value in its respective column. Moreover, ζ_j^+ is the ideal positive solution and ζ_j^- is the ideal negative solution. The ideal positive solution corresponds to the maximum value of z_j^i, while the ideal negative solution corresponds to the minimum value of ζ_j^i ($j = 1, 2, \ldots, k$ and $i = 1, 2, \ldots, m$).

More precisely, the proposed AE-TOPSIS-ECPSA method is a multi-criteria approach for Edge-Cloud environment which considers response time and cost of the Edge/Cloud nodes to guarantee smart decisions by assigning medium to lightweight tasks to resource-constrained edge nodes closer to the user and resource-hungry tasks to cloud nodes with less running cost.

5. Performance evaluation

In this section, the experimental set up and dataset is described and the performance of the proposed methods is evaluated and compared with the state-of-the-art heuristic and ML based algorithms.

Table 4
Configuration of hosts.

Nodes	Host	CPU Model	Cores	Freq. (Hz)	RAM (GB)	Net BW (GB/s)	Disk BW (MB/s)	Cost Model ($/hr)	Power consumption for different loads (W)										
									0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Edge	Hitachi HA 8000	Intel i3 3.0 GHz	2	1800	8	0.1	76	0.11	24.3	30.4	33.7	36.6	39.6	42.2	45.6	51.8	55.7	60.8	63.2
	DEPO Race X340H	Intel i5 3.2 GHz	4	2000	16	1	49	0.23	83.2	88.2	94.3	101	107	112	117	120	124	128	131
Cloud	Dell PowerEdge R820	Intel Xeon 2.6 GHz	32	2000	48	1	49	3.47	110	149	167	188	218	237	268	307	358	414	446
	Dell PowerEdge C6320	Intel Xeon 2.3 GHz	64	2660	64	1.5	1024	6.94	210	371	449	522	589	647	705	802	924	1071	1229

5.1. Experiment setup

Given that our target system operates in a heterogeneous edge-cloud computing environment, it is crucial to account for the varying computational capabilities of both edge and cloud hosts. In our infrastructure setup, an Edge-Cloud computing infrastructure is simulated that includes a data center with 800 heterogeneous physical machines. This consists of 200 Hitachi HA 8000, 200 DEPO Race X340H, 200 Dell PowerEdge R820, and 200 IDell PowerEdge C6320. Characteristics of these machines are outlined in Table 4, and their power consumptions is calculated using the data presented there, following the Edge and Cloud nodes in Tuli et al. (2020). The response time of edge-loud nodes is set 1 ms, while for cloud nodes it is set to 10 ms based on the empirical studies of FogBus edge-cloud framework (Tuli et al., 2019).

We utilize an extended version of Cloudsim toolkit (Calheiros et al., 2011) as our simulation platform, along with the complete infrastructure it provides (Tuli et al., 2020). This toolkit allows us to conduct repeatable experiments on large-scale virtualized Edge-Cloud environment, taking into account factors such as response time, cost and power consumption of edge nodes. Furthermore, CloudSim is an open-source, modular and extensible toolkit that can model the mobility of IoT devices by simulating bandwidth fluctuations and delayed task execution in cloud environments. Our implementation interacts with machine learning software for input pre-processing and AE conversion and new modules are created to simulate PCA and TOPSIS methods.

In the simulation environment, each task in the Edge-Cloud setup is represented by a container that runs until it finishes execution. These Containers are assigned to VMs on physical servers. While each container handles only one task, a VM can host multiple containers.

The dynamic workloads are generated using the real-world open-source Bitbrain's dataset (Shen et al., 2015). This dataset includes real traces of resource consumption metrics from over a thousand VMs on Bitbrain infrastructure with logs recorded at 5-min intervals, detailing requested CPU cores, CPU usage (in MIPS) and RAM characteristics. Moreover, Google dataset (Reiss et al., 2011) collected from a containerized platform and containing task's runtimes and priorities, is used to select container runtimes. By combining Bitbrain and Google datasets, a mixed workloads s created that exhibit different utilization pattern but similar runtimes. This prompts the consolidation module to handle migrations differently, depending on the hosts' utilization levels. The VM types used align with Amazon EC2 instance types, while the container types are modeled after Google machine types.

During the simulations, container are launched dynamically when a request matches task arrival times from the Google cluster dataset. The container runtimes corresponds t the tasks execution times from Google's cluster data. Containers are assigned to VMs, with each VM capable of hosting multiple containers. Once a container completes its task or consumes fewer resources than provisioned, opportunities for consolidation arise.

5.2. Comparison with the state-of-the-art heuristic methods

In this section, our proposed policy is evaluated by comparing it with seven other approaches proposed in the state of the arts, including Arianyan et al. (2015), Piraghaj et al. (2017), Ma et al. (2020), Khan et al. (2020) and Gholipour et al. (2020). Approaches are numbered from one to eight, as depicted in the first column of Table 5. The proposed solution is Approach 8. In the last seven columns of Table 5, an abbreviation name of the policies used for each sub-problem (stage) of the corresponding consolidation approach is reported.

Approach 1 indicates the four stage VM consolidation solution using the optimal algorithms suggested in Arianyan et al. (2015). This includes LR policy for identifying over-loaded hosts, TOPSIS-Available Capacity and Number of VM and migration Delay (TACND) for detecting under-loaded hosts, MMT policy for VM selection, and TOPSIS power and SLA aware allocation (TPSA) for determining new destinations for migrating VMs.

Approach 2 indicates the four stage container consolidation solution using the optimal algorithms suggested in Piraghaj et al. (2017). This includes static threshold (Static THR) policy with 70% threshold for identifying under-loaded host and 80% threshold for identifying over-loaded hosts, Maximum Usage (MU) for migrating containers selection, and correlation host selection policy (CORHS) for container placement. If the correlation of the containers at the destination host is not available, the least full host selection (LFHS) policy is used as an alternative.

Approach 3 indicates container migration-based decision-making approach (Ma et al., 2020) including static threshold policy for both resource utilization and load diversity between hosts for determination of over-loaded hosts, maximum usage (MU) for selection of the containers that should be migrated, and Load Balancing Joint Migration Cost (LBJC) minimization for container placement that minimizes the latency impact of container migration while balancing the load of hosts.

Approach 4 indicates an extension of approach 1 in which the proposed AE-TOPSIS-ECPSA method is used in the fourth VM placement stage. This approach is designed in order to investigate the effect of the proposed AE-TOPSIS-ECPSA policy on consolidation frameworks that are based on only VM migrations.

Approach 5 indicates the six stage consolidation solution allowing both VM and container migrations (Khan et al., 2020) including static threshold policy (Static THR) with 70% threshold for determination of under-loaded host and 80% threshold for determination of over-loaded hosts, Energy and Performance Efficient consolidation (EPC) policy for selection of the containers/VMs for migration, and Modified Best Fit Decreasing (MBFD) for container/VM placement. EPC policy uses the product of energy consumption and performance (reciprocal of the execution time) as a rating metric to prioritize those containers/VMs that enhance energy and performance more.

Approach 6 indicates the seven stage joint VM and container consolidation solution, as proposed in Gholipour et al. (2020). It includes

Table 5
The characteristics of approaches.

Approach Num.	Approach description	Host overload	Host underload	Candidate VM/container for migration	VM selection	VM placement	Container selection	Container placement
1	VM migration in Cloud environment (Arianyan et al., 2015)	LR	TACND	–	MMT	TPSA	–	–
2	Container migration in Cloud environment (Piraghaj et al., 2017)	Static-THR	Static-THR	–	–	–	MU	CORHS, LFHS
3	Container migration in Edge-Cloud environment (Ma et al., 2020)	–	Static-THR	–	–	–	MU	LBJC
4	VM migration in Edge-Cloud environment with AE-TOPSIS based VM placement	LR	TACND	–	MMT	AE-TOPSIS-ECPSA	–	–
5	Joint VM and container migration in Cloud environment (Khan et al., 2020)	Static-THR	Static-THR	–	EPC	MBFD	EPC	MBFD
6	Joint VM and container migration in Cloud environment (Gholipour et al., 2020)	LR	SM	JVCMMD	MMT	MU	TPSA	CORHS, LFHS
7	Joint VM and container migration in Edge-Cloud environment with AE-TOPSIS based VM placement	LR	SM	JVCMMD	MMT	MU	AE-TOPSIS-ECPSA	CORHS, LFHS
8	Joint VM and container migration in Edge-Cloud environment with AE-TOPSIS based VM selection and placement	LR	SM	AE-TOPSIS-JVCMMD	MMT	MU	AE-TOPSIS-ECPSA	CORHS, LFHS

LR for identifying over-loaded hosts, SM for detecting under-loaded hosts, MU policy for selecting containers to migrate, MMT policy for identifying VMs to migrate, JVCMMD policy to decide whether to migrate VMs or containers, CORHS policy for container placement, with LFHS as an its alternative when VM-host correlation data is unavailable, and TPSA policy for determining new placements for VMs.

Approach 7 indicates an extension of approach 6 in which the proposed AE-TOPSIS-ECPSA method is used in the seventh stage to find new destinations for the VMs that should be migrated. This approach is designed in order to investigate the effect of the proposed AE-TOPSIS-ECPSA policy on consolidation frameworks that are based on joint VM and container migrations.

Approach 8 indicates our proposed joint VM and container consolidation solution in this paper similar to scenario 6 in which the proposed AE-TOPSIS-JVMCMMD and AE-TOPSIS-ECPSA policies are used in fifth and seventh stage for migrating VM selection and placement, respectively. The proposed AE-TOPSIS-ECPSA adapts previous TPSA method to Edge-Cloud environment by considering extra criteria including response time and cost of Edge/Cloud nodes. Both proposed AE-TOPSIS-ECPSA and AE-TOPSIS-JVMCMMD methods exploit AE and TOPSIS modules to extract the nonlinear contribution of decision criteria and compute the scores of all the alternatives.

Ten experiments are performed separately for the ten days (Bitbrain+Google dataset) and the results of eight approaches for energy consumption, SLAV, response time, running cost, number of VM migrations and number of container migrations are reported in Table 6. The experiment is then repeated for different workload types from Google's cluster dataset (Reiss et al., 2011) and the results of all experiments corresponding to three workload types W0, W4, and W9 which belong to tasks of three different priorities (0, 4, 9) are shown graphically in Fig. 5.

To determine the statistical significance of the results, we perform a two-sample t-test with a significance level of 0.05. The null hypothesis states that the performance measure of all other approaches is less than or equal to that of the proposed method (Approach 8). The one-tailed p-values from the tests are presented in the experimental results in Table 6.

The results of our proposed solution is compared with the best previous ones in three categories which are based on either only VM migrations (Approaches 1 and 4), only container migrations (Approaches 2 and 3) or joint VM and container migration (Approaches 4 to 8). It can be inferred from Table 6 that Approaches 4 and 7 as extensions of Approaches 1 and 6 using the proposed AE-TOPSIS-ECPSA method, show superior performance in comparison to original methods. Moreover, it can be inferred from Table 6 that adopting our proposed approach (Approach 8) leads to 47%, 63%, 16%, 17.9%, and 62.5%, decrease in energy consumption, SLA violation, response time, cost, and the number of VM migrations, respectively, when compared to the best previous method based solely on VM migrations (Approach 1). Although, approach 4 in Table 6 (as an extension of Approach 1 using the proposed AE-TOPSIS-ECPSA method) shows a negligible less median cost than the proposed method (approach 8), the standard deviations and p-value of t-test indicate no significant difference between medians and the cost of the two methods is then comparable.

Also, it can be inferred from Table 6 that adopting our proposed approach (Approach 8) results in 56.9%, 67.3%, 18.6%, 31.25%, 9.65% decrease in energy consumption, SLA violation, response time, cost, and the number of container migrations, respectively, when compared

Table 6
Performance comparison between the proposed method (Approach 8) and seven other heuristic consolidation solutions. The median performance and corresponding standard deviation are reported (One-tailed p-values from two sample t-tests, testing the null hypothesis that the performance metrics of Approach 8 exceeds that of the other approaches are provided in parenthesis).

Approach num.	Energy consumption (x10 kW)	Response time (ms)	Cost (US $)	SLAV	# VM migration (%)	# container migration (%)
1	764.96 ± 51 (0.0000)	8.8 ± 0.6 (0.0000)	7125 ± 452 (0.0000)	0.0138 ± 0.001 (0.0000)	2.25 ± 0.14 (0.0000)	0 ± 0 (0.0000)
2	934.22 ± 63 (0.0000)	9.1 ± 0.5 (0.0000)	8505 ± 561 (0.0000)	0.0156 ± 0.001 (0.0000)	0	5.4 ± 0.34 (0.0000)
3	826.22 ± 58 (0.0000)	8.9 ± 0.5 (0.0000)	8125 ± 481 (0.0000)	0.0156 ± 0.001 (0.0000)	0	4.7 ± 0.32 (0.0000)
4	661.56 ± 42 (0.0000)	7.8 ± 0.5 (0.0000)	**5640 ± 373** (0.002)	0.0112 ± 0.001 (0.0000)	2.07 ± 0.13 (0.0000)	0 ± 0 (0.0000)
5	740.32 ± 45 (0.0000)	8.2 ± 0.6 (0.0000)	6740 ± 434 (0.0000)	0.011 ± 0.001 (0.0000)	2 ± 0.13 (0.0000)	4.42 ± 0.29 (0.0000)
6	688.66 ± 45 (0.0000)	8.5 ± 0.6 (0.0000)	6523 ± 434 (0.0000)	0.0073 ± 0.001 (0.0000)	2.02 ± 0.12 (0.0000)	4.18 ± 0.27 (0.0000)
7	581.44 ± 38 (0.0012)	7.6 ± 0.5 (0.0023)	6250 ± 314 (0.0065)	0.0061 ± 0.000 (0.00765)	1.28 ± 0.009 (0.0231)	3.5 ± 0.23 (0.0098)
8	**402.57 ± 26**	**7.4 ± 0.4**	**5847 ± 388**	**0.0051 ± 0.000**	**0.84 ± 0.05**	**1.8 ± 0.10**

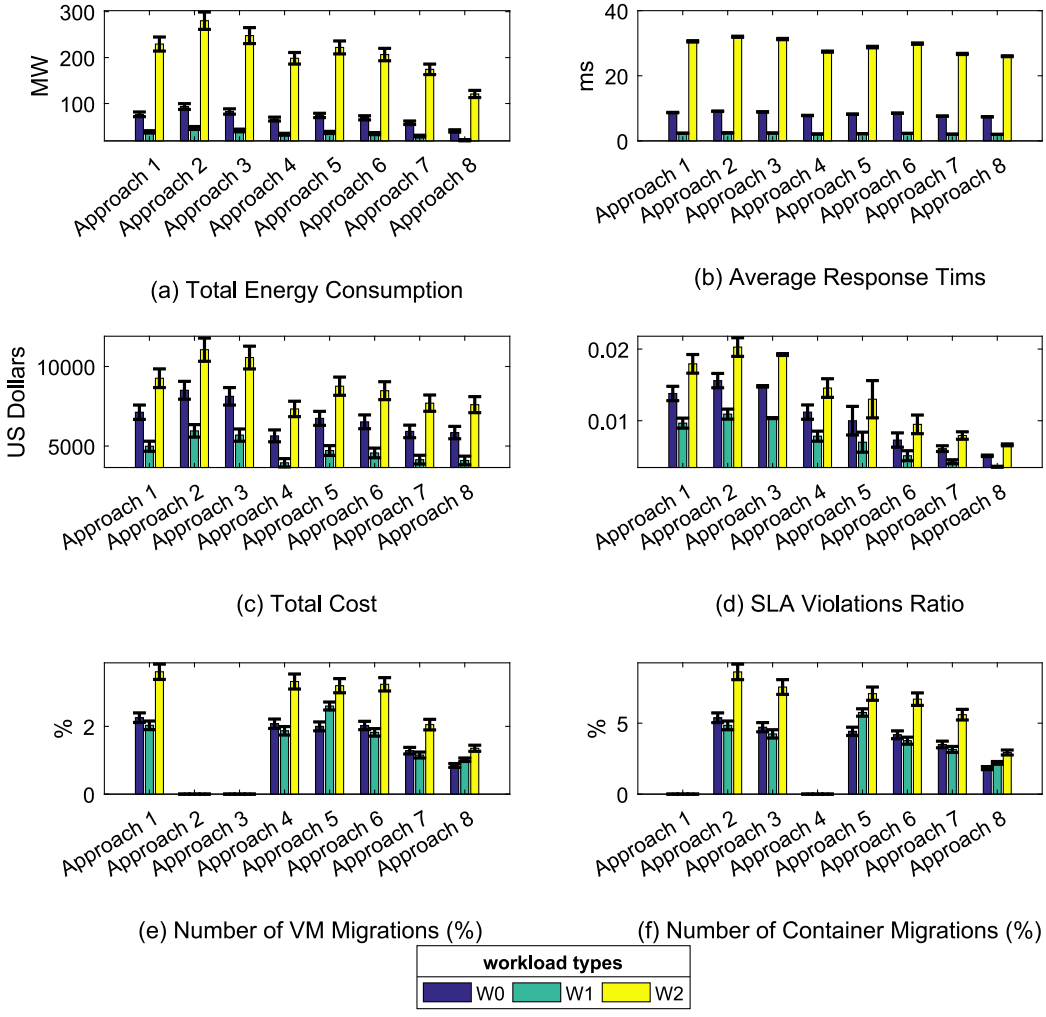

Fig. 5. Comparison of the proposed method with other approaches for different workload types.

to the best previous methods based solely on container migrations (Approach 2 and Approach 3).

Furthermore, the results in Table 6 show that adoption of our proposed approach (Approach 8) leads to 41.5%, 30.13%, 12.9%, 10.3%, 58.2% and 56.1% decrease in energy consumption, SLA violation, response time, cost, number of VM migrations, and number of container migrations, respectively, in comparison with the best previous approach based on joint VM and container migrations (Approach 5 and Approach 6).

From Fig. 5, it can be seen that for all Monitoring/Console workloads (W0), short-running workloads (W4), and long-running workloads (W9) the proposed method shows superior performance in comparison to other approaches. Workloads W4 are characterized by their quick execution times and frequent start/stop cycles. Consolidation approaches reduce energy consumption by optimizing the placement of these tasks to minimize the number of active nodes. Although, inaccurate prediction of runtime in approach 5 potentially increases its total number of migrations, our optimal resource management strategy can mitigate this effect to maintain number of migrations while minimizing energy usage. Workloads W9 require consistent performance over extended periods. Consolidation for these tasks requires balancing energy efficiency with performance guarantees. The proposed approach successfully consolidate these tasks in a way that avoid performance degradation without very excessive migrations in comparison to other approaches.

So, the obtained results validate the applicability of our proposed approach for consolidation in Edge-Cloud environment. This observation can be described by the fact that our proposed AE-TOPSIS-ECPSA policy includes extra criteria including response time and cost of Edge/Cloud nodes in comparison to the previous TPSA method. It ensures to make smart decisions by assigning resource-hungry tasks to resource-abundant cloud nodes and medium to lightweight tasks to resource-constrained edge nodes closer to the user. Moreover, our proposed joint VM and container consolidation solution takes advantage of the AE and TOPSIS modules to extract the nonlinear contribution of decision criteria in both proposed AE-TOPSIS-ECPSA and AE-TOPSIS-JVCMMD policies. It successfully trades off between our goals, including energy consumption, cost, response time, SLA violation and the number of migrations.

5.3. Comparison with the state-of-the-art ML-based methods

In this section the performance of our proposed model is compared with the results of ML-based methods reported in Tuli et al. (2020), namely

1. DDQN: Double variant of standard Deep Q-Network similar to the experiment in literature (Basu et al., 2019; Li and Hu, 2019).
2. DRL (REINFORCE): A deep reinforcement learning method using policy gradient and a fully connected neural network (Mao et al., 2016).
3. An asynchronous policy gradient reinforcement learning approach known as Asynchronous Advantage Actor Critic which utilizes a Residual Recurrent Neural Network (A3C-R2N2) (Tuli et al., 2020).

The graphs of DDQN and REINFORCE and A3C-R2N2 in Fig. 6 are reproduced using the results reported in Tuli et al. (2020). It is important to note that the above three methods only consider VM migrations. In order to be able to compare the results obtained by the proposed method with the ones obtained by other ML-based methods, the results of AE-TOPSIS-ECPCA (approach 4 in Table 5), as a simplified version of the proposed method that only considers VM migration is also reported. The last bar of Fig. 6 represents the results of the proposed joint VM and container migration method (approach 8 in Table 5). The results are obtained in an experiment similar to one reported in Tuli et al. (2020) and the metrics are computed exactly as reported in it. Settings of the methods allowing VM migration alone include a one-to-one allocation from tasks to VMs and when the allocated task is completed, the corresponding VM is discarded (Tuli et al., 2020). For the last method setting, each task represent a container that runs until its execution is completed. Containers are allocated to VMs and each VM can host more than one container (task) with similar runtime to other methods. In all studied methods, opportunities for consolidation are created when container/VM finishes the allocated task, or task utilizes the provisioned resources less.

Fig. 6(a)(c) shows that the lowest cost and energy consumption are made by the proposed AE-TOPSIS-JVCMMD method and AE-TOPSIS-ECPSA approach, as a simplified version of AE-TOPSIS-JVCMMD without container migration capability in this paper. This is due to their novel structure which allows simultaneous Edge-Cloud environment sensing (via power and cost of edge nodes) and non-linear contribution capturing of input criteria.

By investigating Fig. 6(b)(d), the proposed AE-TOPSIS-ECPSA approach shows a negligible more mean response time and SLA violations than A3C-R2N2, but overlapping error bars indicate no significant difference between means and the performance of the two methods is then comparable. Moreover, the SLA violation and the response time in the proposed AE-TOPSIS-JVCMMD method are significantly less than all other ML-based methods. Also, as seen in Fig. 6(e), the proposed model achieves the highest number of completed tasks. Since the frequency and duration of migrations significantly impact task response time, Fig. 6(f) show how the proposed model attains the best metrics by minimizing the number of VM migrations. This can be explained by the fact that our proposed joint VM and container consolidation solution employs AE-TOPSIS-JVCMMD policy to decide whether to migrate VMs or containers. Moreover, AE-TOPSIS-JVCMMD policy leverages both VM and container migration to consolidate VMs, ensuring that if no suitable host is found for the VMs selected for migration, their containers are migrated instead, leading to a significant improvements in results.

Moreover, the developed methods that train an end-to-end network with fixed output layer structure, face problem of limited scalability or can schedule for a fixed number of edge nodes and tasks.

5.4. Complexity and scalability

The complexity of the proposed method depends on multiple tasks. The maximum time complexity of TOPSIS algorithm is O(nk) which results from the calculation of attribute normalization and weighting. The complexity of PCA is O(dn2+d3), where the two terms correspond to covariance matrix computation and eigen-value decomposition, respectively. As PCA is only used for the first initialization phase and then Autoencoders (AE) are utilized to extract input criteria contributions, and their forward pass and backpropagation steps can be ignored since they are executed on Graphics Processing Units (GPUs). For N scheduling intervals, the overall time complexity is O(nkN), making the computational cost similar to that of machine learning-based methods.

6. Concluding remarks and future directions

This paper proposes an efficient consolidation of VM and containers in edge-cloud environment defined by a variety of virtualization technologies, resource heterogeneity, and varying response times between devices in the Edge and Cloud layers. Previous research overlooks these distinctions between edge and cloud devices and neglects the interdependence of decision criteria, which leads to a biased preference for options that perform well in two or more related criteria.

More specifically, this paper focuses on the standard seven-phase joint VM and consolidation solution and proposes two multi-criteria decision making policies for migrating VMs/containers selection and

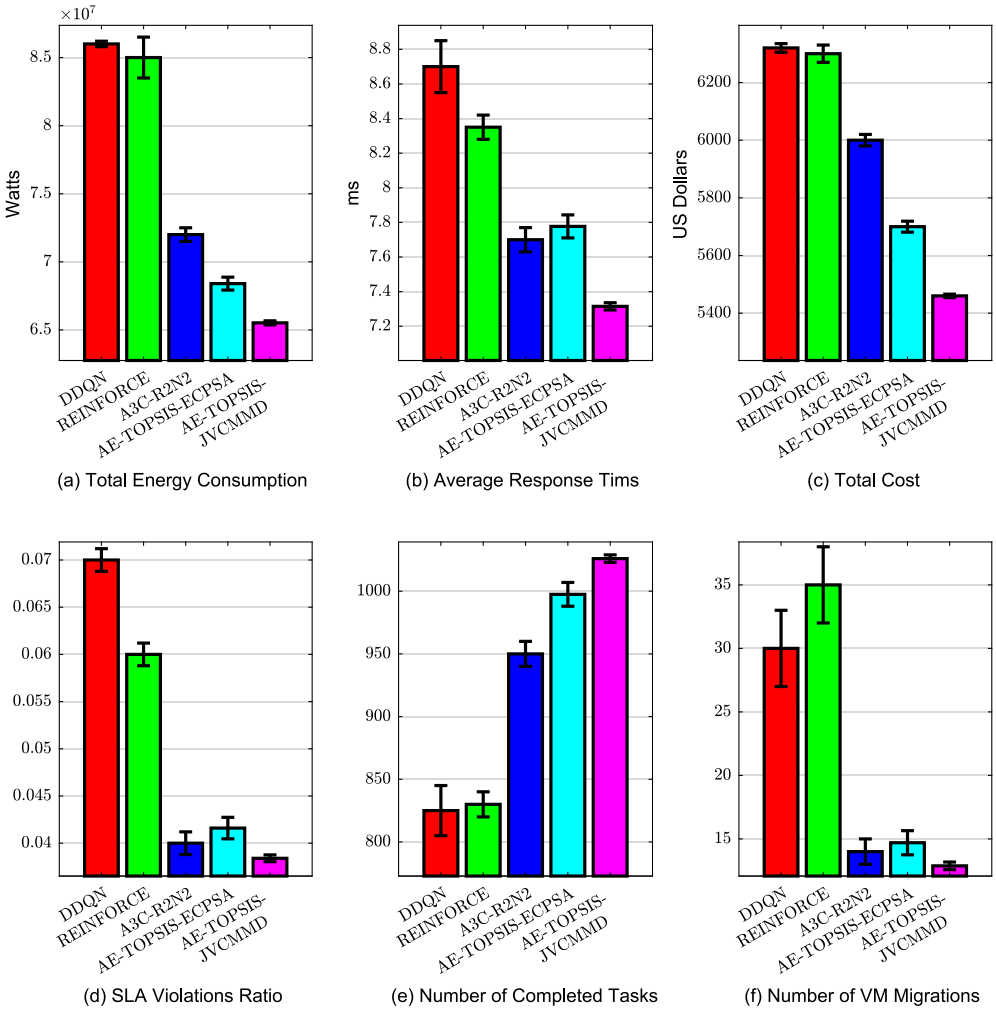

Fig. 6. Comparison of the proposed method with ML-based approaches.

placement, namely AE-TOPSIS-JVCMMD and AE-TOPSIS-ECPSA in the fifth and the seventh phases, respectively.

The proposed AE-TOPSIS-ECPSA method is the customized multi-criteria approach for Edge-Cloud environment which considers extra criteria, namely response time and cost of the Edge/Cloud nodes to guarantee smart decisions by assigning resource-hungry tasks to resource-abundant cloud nodes and lightweight tasks to resource-constrained edge nodes closer to the user. Moreover, both proposed AE-TOPSIS-ECPSA and AE-TOPSIS-JVCMMD policies takes advantage of the AE-TOPSIS module to extract the nonlinear contribution of decision criteria and avoid the bias in the calculation of the ideal solutions due to the correlation between criteria.

Experimental results show that adoption of our solution results in 41.5%, 30.13%, 12.9%, 10.3%, 58.2% and 56.1% decrease in energy consumption, SLA violation, response time, cost, number of VM migrations, and number of container migrations, respectively, in comparison with the standard joint VM and container consolidation approach. It successfully makes trade-off between the main goals, including energy consumption, SLA violation, and the number of migrations.

As part of future work, we plan to integrate the proposed algorithms with Kubernetes (Ellingwood et al., 2016) and its extensions for the Edge-Cloud environments (Cappos et al., 2018), thanks to Kubernetes which natively supports replacing the default scheduler of each cluster with a custom one. Our framework allows the clusters to receive consolidation decisions from a centralized control plane in order to make in-cluster scheduling, pod autoscaling, and cluster autoscaling decisions. Another research direction is considering other variants of AE nonlinear contribution extraction, namely regularized or variational AEs rather than standard structure.

CRediT authorship contribution statement

Farkhondeh Kiaee: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology. **Ehsan Arianyan:** Validation, Supervision, Project administration.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Farkhondeh Kiaee reports financial support was provided by Iran National Science Foundation (INSF). Farkhondeh Kiaee reports equipment, drugs, or supplies was provided by ICT Research Institute (ITRC).

Acknowledgments

The authors would like to acknowledge the technical support by the ICT Research Institute (ITRC), Iran. This work is based upon research funded by Iran National Science Foundation (INSF), Iran under project No. 4004202.

Data availability

Data will be made available on request.

References

Arianyan, E., Taheri, H., Sharifian, S., 2015. Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. Comput. Electr. Eng. 47, 222–240.

Basu, D., Wang, X., Hong, Y., Chen, H., Bressan, S., 2019. Learn-as-you-go with megh: Efficient live migration of virtual machines. IEEE Trans. Parallel Distrib. Syst. 30 (8), 1786–1801.

Beloglazov, A., Buyya, R., 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput.: Pract. Exper. 24 (13), 1397–1420.

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. - Pract. Exp. 41 (1), 23–50.

Cappos, J., Hemmings, M., McGeer, R., Rafetseder, A., Ricart, G., 2018. EdgeNet: A global cloud that spreads by local action. In: 2018 IEEE/ACM Symposium on Edge Computing. SEC, IEEE, pp. 359–360.

Chen, T., Zhu, Y., Gao, X., Kong, L., Chen, G., Wang, Y., 2018. Improving resource utilization via virtual machine placement in data center networks. Mob. Netw. Appl. 23 (2), 227–238.

Du, M., Li, F., 2017. ATOM: efficient tracking, monitoring, and orchestration of cloud resources. IEEE Trans. Parallel Distrib. Syst. 28 (8), 2172–2189.

Ellingwood, J., et al., 2016. An introduction to kubernetes. Digit. Ocean 14.

Feng, C., Han, P., Zhang, X., Yang, B., Liu, Y., Guo, L., 2022. Computation offloading in mobile edge computing networks: A survey. J. Netw. Comput. Appl. 103366.

Fu, K., Zhang, W., Chen, Q., Zeng, D., Guo, M., 2021. Adaptive resource efficient microservice deployment in cloud-edge continuum. IEEE Trans. Parallel Distrib. Syst. 33 (8), 1825–1840.

Garí, Y., Monge, D.A., Pacini, E., Mateos, C., Garino, C.G., 2021. Reinforcement learning-based application autoscaling in the cloud: A survey. Eng. Appl. Artif. Intell. 102, 104288.

Ghobaei-Arani, M., Souri, A., Rahmanian, A.A., 2020. Resource management approaches in fog computing: a comprehensive review. J. Grid Comput. 18 (1), 1–42.

Gholipour, N., Arianyan, E., Buyya, R., 2020. A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers. Simul. Model. Pract. Theory 104, 102127.

Gholipour, N., Arianyan, E., Buyya, R., 2022. Recent advances in energy-efficient resource management techniques in cloud computing environments. In: New Frontiers in Cloud Computing and Internet of Things. Springer, pp. 31–68.

Horri, A., Mozafari, M.S., Dastghaibyfard, G., 2014. Novel resource allocation algorithms to performance and energy efficiency in cloud computing. J. Supercomput. 69 (3), 1445–1461.

Jiang, H.-P., Chen, W.-M., 2018. Self-adaptive resource allocation for energy-aware virtual machine placement in dynamic computing cloud. J. Netw. Comput. Appl. 120, 119–129.

Khan, A.A., Zakarya, M., Khan, R., Rahman, I.U., Khan, M., et al., 2020. An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters. J. Netw. Comput. Appl. 150, 102497.

Lebre, A., Pastor, J., Simonet, A., Südholt, M., 2018. Putting the next 500 vm placement algorithms to the acid test: The infrastructure provider viewpoint. IEEE Trans. Parallel Distrib. Syst. 30 (1), 204–217.

Li, F., Hu, B., 2019. Deepjs: Job scheduling based on deep reinforcement learning in cloud data center. In: Proceedings of the 2019 4th International Conference on Big Data and Computing. pp. 48–53.

Li, C., Tang, J., Luo, Y., 2020. Elastic edge cloud resource management based on horizontal and vertical scaling. J. Supercomput. 76, 7707–7732.

Li, W., Yue, H.H., Valle-Cervantes, S., Qin, S.J., 2000. Recursive PCA for adaptive process monitoring. J. Process Control 10 (5), 471–486.

Ma, Z., Shao, S., Guo, S., Wang, Z., Qi, F., Xiong, A., 2020. Container migration mechanism for load balancing in edge network under power Internet of Things. IEEE Access 8, 118405–118416.

Maenhaut, P.-J., Volckaert, B., Ongenae, V., De Turck, F., 2020. Resource management in a containerized cloud: Status and challenges. J. Netw. Syst. Manage. 28 (2), 197–246.

Manimurugan, S., 2021. IoT-Fog-Cloud model for anomaly detection using improved Naive Bayes and principal component analysis. J. Ambient Intell. Humaniz. Comput. 1–10.

Mao, H., Alizadeh, M., Menache, I., Kandula, S., 2016. Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. pp. 50–56.

Mateos, C., Hirsch, M., Toloza, J.M., Zunino, A., 2022. LiveDewStream: A stream processing platform for running in-lab distributed deep learning inferences on smartphone clusters at the edge. SoftwareX 20, 101268.

Pelegrina, G.D., Duarte, L.T., Romano, J.M.T., 2019. Application of independent component analysis and TOPSIS to deal with dependent criteria in multicriteria decision problems. Expert Syst. Appl. 122, 262–280.

Pham, X.-Q., Huh, E.-N., 2016. Towards task scheduling in a cloud-fog computing system. In: 2016 18th Asia-Pacific Network Operations and Management Symposium. APNOMS, IEEE, pp. 1–4.

Piraghaj, S.F., Dastjerdi, A.V., Calheiros, R.N., Buyya, R., 2017. Containercloudsim: An environment for modeling and simulation of containers in cloud data centers. Softw. - Pract. Exp. 47 (4), 505–521.

Reiss, C., Wilkes, J., Hellerstein, J., 2011. Google Cluster-Usage Traces: Format+ Schema. Google Inc., Mountain View, CA, USA.

Seuret, M., Alberti, M., Liwicki, M., Ingold, R., 2017. PCA-initialized deep neural networks applied to document image analysis. In: 2017 14th IAPR International Conference on Document Analysis and Recognition. ICDAR, Vol. 1, IEEE, pp. 877–882.

Shen, S., Van Beek, V., Iosup, A., 2015. Statistical characterization of business-critical workloads hosted in cloud datacenters. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, pp. 465–474.

Soni, D., Kumar, N., 2022. Machine learning techniques in emerging cloud computing integrated paradigms: A survey and taxonomy. J. Netw. Comput. Appl. 103419.

Tuli, S., Ilager, S., Ramamohanarao, K., Buyya, R., 2020. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. IEEE Trans. Mob. Comput. 21 (3), 940–954.

Tuli, S., Mahmud, R., Tuli, S., Buyya, R., 2019. Fogbus: A blockchain-based lightweight framework for edge and fog computing. J. Syst. Softw. 154, 22–36.

Wang, Z., Goudarzi, M., Aryal, J., Buyya, R., 2023a. Container orchestration in edge and fog computing environments for real-time iot applications. In: Computational Intelligence and Data Analytics. Springer, pp. 1–21.

Wang, W., Zhang, Y., Huang, R., Ren, J., Lyu, F., Zhang, Y., 2023b. Efficient resource management and expansion scheme for collaborative edge-cloud computing. IEEE Trans. Mob. Comput..

Wu, Q., Ishikawa, F., Zhu, Q., Xia, Y., 2016. Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. IEEE Trans. Serv. Comput. 12 (4), 550–563.

Xu, C., Jiang, S., Luo, G., Sun, G., An, N., Huang, G., Liu, X., 2020. The case for fpga-based edge computing. IEEE Trans. Mob. Comput. 21 (7), 2610–2619.

Zakarya, M., Gillam, L., Salah, K., Rana, O., Tirunagari, S., Buyya, R., 2022. CoLocateMe: Aggregation-based, energy, performance and cost aware VM placement and consolidation in heterogeneous IaaS clouds. IEEE Trans. Serv. Comput..

Farkhondeh Kiaee received the Ph.D. degree in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran in 2015 before being postdoctoral fellow at Computer Vision and Systems Laboratory (CVSL) at Université Laval, Quebec City, Canada. She is now an assistant professor at the Department of Electrical Engineering of National University of Skills (NUS). Her research interests include machine learning, fog computing, and Internet of Things (IoT).

Ehsan Arianyan received the B.Sc. degree from Iran University of Science and Technology, Tehran, Iran, in 2008. Also, he received his M.Sc. and PhD. degrees from Amirkabir University of Technology, Tehran, Iran, in 2010 and 2015, respectively. He is now an assistant professor of ICT Research Institute (ITRC) and working as the head of IT Faculty. He is the author of more than 20 peer-reviewed papers as well as 10 books related to cloud computing. His research interests include cloud computing, parallel computing, big data, and decision algorithms.