Journal of Grid Computing (2024) 22:6
https://doi.org/10.1007/s10723-023-09727-1

RESEARCH

Anomaly Detection in Cloud Computing using Knowledge Graph Embedding and Machine Learning Mechanisms

Katerina Mitropoulou · Panagiotis Kokkinos · Polyzois Soumplis · Emmanouel Varvarigos

Received: 19 September 2023 / Accepted: 16 December 2023 / Published online: 29 December 2023

Abstract The orchestration of cloud computing infrastructures is challenging, considering the number, heterogeneity and dynamicity of the involved resources, along with the highly distributed nature of the applications that use them for computation and storage. Evidently, the volume of relevant monitoring data can be significant, and the ability to collect, analyze, and act on this data in real time is critical for the infrastructure's efficient use. In this study, we introduce a novel methodology that adeptly manages the diverse, dynamic, and voluminous nature of cloud resources and the applications that they support. We use knowledge graphs to represent computing and storage resources and illustrate the relationships between them and the applications that utilize them. We then train GraphSAGE to acquire vector-based representations of the infrastructures' properties, while preserving the structural properties of the graph. These are efficiently provided as input to two unsupervised machine learning algorithms, namely CBLOF and Isolation Forest, for the detection of storage and computing overusage events, where CBLOF demonstrates better performance across all our evaluation metrics. Following the detection of such events, we have also developed appropriate re-optimization mechanisms that ensure the performance of the served applications. Evaluated in a simulated environment, our methods demonstrate a significant advancement in anomaly detection and infrastructure optimization. The results underscore the potential of this closed-loop operation in dynamically adapting to the evolving demands of cloud infrastructures. By integrating data representation and machine learning methods with proactive management strategies, this research contributes substantially to the field of cloud computing, offering a scalable, intelligent solution for modern cloud infrastructures.

Keywords Cloud computing · Resource allocation · Knowledge graphs · Anomaly detection

1 Introduction

Most applications and services utilize centralized computing infrastructures, rendering cloud computing a crucial component of our digital economy. These appli-

K. Mitropoulou (✉) · P. Kokkinos · P. Soumplis · E. Varvarigos
Department of Electrical & Computer Engineering, National Technical University of Athens, 9, Iroon Polytechniou St, Athens 15780, Attica, Greece

K. Mitropoulou
e-mail: kmetropoulou@mail.ntua.gr

P. Kokkinos
e-mail: p.kokkinos@uop.gr

P. Soumplis
e-mail: soumplis@mail.ntua.gr

E. Varvarigos
e-mail: vmanos@mail.ntua.gr

P. Kokkinos
Department of Digital Systems, University of Peloponnese, Kladas, Lakoni, 23100, Sparta, Greece

cations relate to smart cities [1], autonomous driving [2], augmented reality [3] and Internet of (IoT), in general. The aforementioned applications are also recognized as critical data generating sources. The number of Internet of Things devices in a wide array of domains, operating in the edge of the network, is forecast to grow to almost 6.5 billion by 2030, an increase of more than 4 billion compared to 2020 [4].

However, utilizing central cloud to transfer and process edge-generated data is deemed impractical due to economic factors, such as the costs of cloud and networking resources, and performance factors, such as high and fluctuating latency and limited throughput [5,6]. Moreover, it has been acknowledged that only a fraction of this data will be actually meaningful or helpful (the edge could act as a filter on what reaches the core cloud) and that their size will surpass the storage capacities of today's cloud data centers [7]. To this end, edge computing has gained ground as a computing paradigm to boost application and infrastructure efficiency by using computing and storage resources close to where data are generated [8]. Edge resources can operate in different layers: on-premise, micro, deep and far edge, providing various levels performance efficiency.

Today, it is recognized that the joint use of edge and cloud resources, in the form of the so called edge-cloud continuum, can provide the most benefits for serving distributed applications while utilizing heterogeneous computing and storage resources. Also, the complexity and dynamicity of the applications' requirements (e.g., in terms of resource usage or delay sensitivity) makes necessary the coupling of resource orchestration mechanisms with advanced monitoring capabilities in a continuous closed-loop control fashion that will run over an infinite time horizon [9]. Figure 1 portrays such a complex cloud computing infrastructure that spans from local IoT devices and edge resources to the expansive core cloud datacenters, conveying the layered approach to data processing and application support within a distributed cloud environment. The effective monitoring of such complex infrastructures can enable proactive and reactive re-optimization operations for the benefit of both the running applications and the infrastructures themselves.

In this work, we use knowledge graphs to represent computing and storage infrastructures and illustrate relationships between the available resources and the applications that utilize them. Leveraging a Knowledge Graph (KG) to model the computing and storage infrastructure is pivotal towards the integration of large datasets of monitoring data, from multiple sources (providers, edge, cloud and network resources, applications etc). This enables obtaining better insights on the infrastructure's status and more efficient decision-making, by performing machine learning based analytics, in comparison to other relevant machine learning applications that only look at the individual nodes' properties in isolation, without taking into account their in-between relations. Next, we use graph embeddings to transform the graph entities (nodes, edges) into fixed length vectors, representing in this way the graph in a low-dimensional space, while preserving its structural properties. We then employ unsupervised anomaly detection methods on the generated embeddings to identify abnormal resource usage patterns, which are often indicative of dynamic or unpredictable application behavior, cybersecurity incidents, and/or bad allocation decisions. This methodology is coupled with appropriate mechanisms that re-optimize the infrastructure usage in a timely manner, by reallocating the applications' workload to other resources.

The remainder of this work is organized as follows. In Section 2 we discuss previous work in the context of automated resource orchestration. In Section 3 we thoroughly describe the proposed closed-loop methodology, discussing the use of knowledge graphs, the application of machine learning mechanisms for event detection and the re-optimization methods applied. In Section 4 we present the experimental setup and the evaluation results of our evaluation. Finally, in Sections 5 and 6 we discuss the limitations and the added value of our approach, and in Section 7 we conclude our work.

2 Related Work

The orchestration of computing and storage resources is a key operation for IT organizations to accelerate service delivery, simplify optimization, and decrease costs [10]. It is also a challenge, considering the heterogeneity and diversity of resource types (CPUs, GPUs, DPUs, storage, bandwidth), the user access patterns, and lifecycle activities [11]. Comprehensive resource allocation techniques capable of spanning across various planes of the processing infrastructure are essential. In particular, AI-based autonomous resource orchestra-

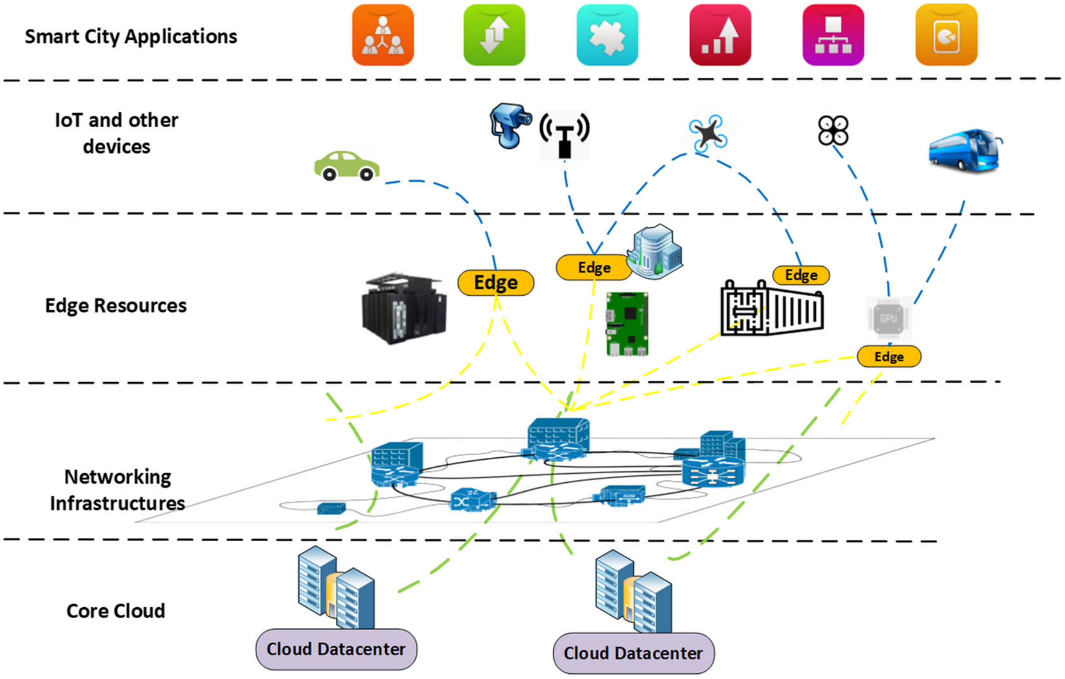


Fig. 1 The distributed nature of modern cloud-based networks, from local devices and edge computing nodes to centralized cloud datacenters

tion mechanisms have been widely explored, leveraging many traditional machine learning (ML) schemes for workload modeling and prediction, load balancing, application placement and migration, computation offloading, system autoscaling, and remediation [12]. Recent studies have introduced agent-based models for cloud simulations, focusing on service placement and migration strategies in various cloud environments [13], and the implementation of recurrent neural networks for dynamic resource allocation [14].

Anomaly detection in resource allocation has seen both supervised and unsupervised ML models being proposed [15–17]. These approaches typically rely on logs or audit traces, including CPU and memory consumption, disk access, and network activity information [18]. Supervised approaches classify anomalous services by training on a labeled dataset, employing methods like decision trees, Random Forests [19,20], and Support Vector Machines (SVM) [21,22]. Ensemble approaches and Self Organizing Maps (SOM) are also used in this context [23,24]. The recent introduction of machine learning-based frameworks for channel bandwidth allocation in distributed computing environments signifies a shift towards more adaptive resource management methods [25].

Unsupervised approaches, which do not rely on annotated data, identify observations that deviate markedly from what is considered normal behavior. Recent advances include the use of recurrent neural networks, autoencoders [26], regression models [27], and clustering algorithms [28]. Timeseries-based algorithms have been used to identify latent correlations in resource usage measurements [29], with statistical correlation analysis used in virtual network functions chains [30]. The introduction of Deep Reinforcement Learning (DRL) approaches in task offloading and resource allocation in edge-cloud networks [31], and in device-to-device (D2D)-aided fog radio access networks (F-RANs) [32], aligns with the trend towards more autonomous resource management systems.

Recent advancements in artificial intelligence to IT operations (AIOps) and cloud resource management have further expanded the scope of intelligent resource orchestration. A study on Prometheus and AIOps highlights their role in automating and optimizing cloud-native applications, pointing towards the future of IT automation that combines DevOps, Big Data, and AI [33]. Another significant development is the use of AIOps for effective cloud resource management to maintain service quality and overall system availability [34]. The dynamic nature of workloads in cloud computing has led to innovative approaches in resource allocation, aiming to ensure Quality-of-Service while managing resource costs [35]. Workload-aware resource management strategies, such as WarMops, focus on optimizing resource allocation and utilization in IaaS clouds, addressing scalability and cost issues [36]. Moreover, the application of deep reinforcement learning in cloud resource scheduling is a promising approach to solving complex combinatorial optimization problems in cloud management [37]. These recent studies underscore the ongoing evolution and complexity in cloud resource management techniques.

Autonomous and intelligent knowledge extraction mechanisms are required to learn and model the dynamic behavior of computing and storage resources, capturing the interactions among system components (e.g., micro-services, edge, cloud, and network resources). KGs are ideal for integrating multi-modal data from heterogeneous sources, enabling natural representation and reasoning about data relationships [38]. They have been used in several fields, such as health [39], social networks [40], recommender systems [41], and cybersecurity [42]. The use of KGs for modeling infrastructures and employing ML-based anomaly detection algorithms, as proposed in our work, remains novel. This approach, coupled with our focus on inductive representation learning for scalable capture of large, complex network topologies, differentiates our methodology from existing studies. Additionally, the cloud failure prediction based on traditional ML and deep learning algorithms [43], and the resource allocation strategies in collaborative cloud-edge computing systems [44], provide further context to the evolving landscape of cloud computing and resource management, affirming the uniqueness and relevance of our research.

Despite the aforementioned adoption of KG approaches , most optimal resource allocation problems are typically modeled as graph problems [45,46], and they are usually solved using queuing theory [47], Q-learning [48], or traditional ML methods [49]. The combined use of graph-based representations and ML techniques towards automated resource allocation is mostly unexplored, with a few notable exceptions that leverage edge graph neural networks for the construction of allocation policies [50] and the prediction of corresponding Quality of Service (QoS) performance indicators [51]. To the best of our knowledge, there is no previous work that combines knowledge graphs and graph embeddings for the modelling of the infrastructures with ML-based anomaly detection algorithms. Moreover, by adopting an inductive representation learning approach that focuses only on the local neighbourhood of each node to extract meaningful embeddings, we can efficiently capture an infrastructure's topology without facing scalability issues even in cases involving large, complex networks.

3 Knowledge Graph-Based Anomaly Detection and Resource Allocation

The proposed methodology materializes a ML-driven pipeline composed of five high-level steps, as depicted in Fig. 2. The pipeline begins by representing the infrastructure entities and their interrelationships as a knowledge graph. Topological embeddings are then extracted from this graph to serve as features for a set of unsupervised methods. This allows us to capitalize on the ability to learn from both the topological structure of a node's neighborhood and the distribution of node features within it.

Currently, the two types of events that our framework can detect are computing and storage overusage events, which are largely identified based on the distribution of node features. More specifically, a computing overusage event arises when the total computational power requested by all applications served on a particular node surpasses the node's available computational resources. Similarly, a storage overusage event occurs when the total storage requested by these applications exceeds the node's storage capacity.

The resource overusage events may be due to bad resource allocation decisions, to malicious behaviour from applications or other reasons. However, it is worth noting that our graph-structured embeddings, which capture the local neighbourhood topology, open up the possibility of detecting additional types of anomalies. For example, an application unexpectedly spanning across multiple resource nodes, potentially indicative of a spamming event, could be perceived as an anomaly due to the unusual topological configuration it presents. While this scenario is beyond the scope of the present

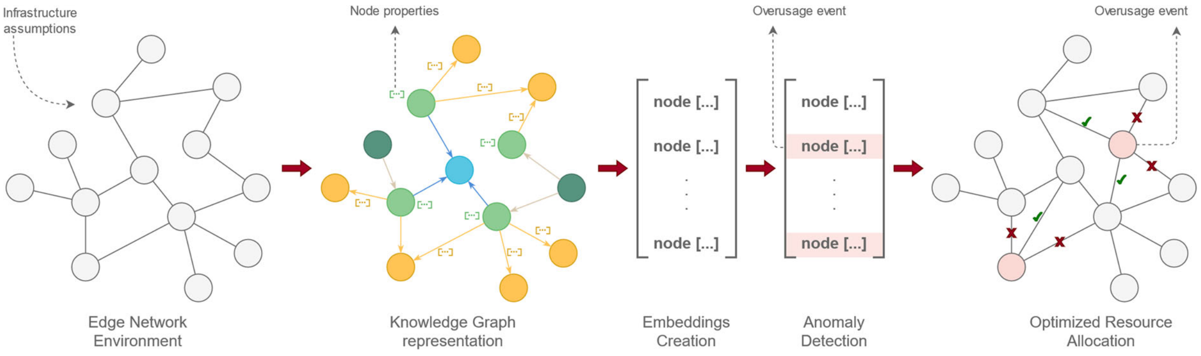


Fig. 2 Overview of the proposed KG-based modelling and event detection methodology

paper, our framework could be extended to identify such topology-based anomalies in future work.

Unsupervised algorithms operate autonomously, as they do not require manual input (e.g., data labelling) from the user, which would be very difficult to obtain anyway. Upon detection of an anomaly, a resource re-optimization algorithm is activated. This algorithm identifies the affected applications and mitigates the sources of the anomalies, for instance by migrating workload to other resources.

Recurring inference, a feature provided by our implemented pipeline, is also utilized to facilitate periodic retraining and to trigger the resource optimization mechanism, enabling continuous control. In this way, we formulate a continuous, autonomous, and closed-loop control system that serves as an anomaly detector and re-optimizer within a computing and storage infrastructure.

3.1 Knowledge-Graph for the Cloud

We consider a computing and storage infrastructure consisting of four types of entities: region entities, provider entities, resource entities and application entities. The following assumptions are made regarding these entities:

- A provider owns a number of resources and each resource belongs to a single provider.
- Each region contains many resources from different providers and each resource is located to a particular region.
- A resource provides computing or storage capacity to multiple applications that share the respective resources.
- The applications consist of multiple components operating in a distributed manner. Each application's components can be deployed in up to five different resources based on their requirements and the resources' capabilities.

Based on the provided infrastructure a knowledge graph is populated, which can be queried using the Cypher query language [52,53]. Each entity of the considered model is assigned to a different node type. Also, the relations between the various entities described earlier are represented in the graph as different types of edges that connect the various types of nodes. A region entity is connected with a resource entity via an edge of type ”located in”, if the resource is located to the particular region. A provider entity is connected with a resource entity via an edge of type ”belongs to”, if the resource belongs to that particular provider. Finally, a resource entity is connected with an application entity via an edge of type ”runs in”, in case a component of the application runs in that resource. This edge also carries a set of weights that represent the percentage of the resource's computing and storage capacity utilized by the application's component(s).

The knowledge graph that represents all node labels and relationship types is presented in Fig. 3, while a snapshot of it is illustrated in Fig. 4.

3.2 Creation of Graph Embeddings

The graph generated at the previous phase provides an intuitive way to depict the infrastructure. However, applying ML operations (e.g., anomaly detection) directly on the graph structure can be a challenging process as most ML algorithms are designed to work with

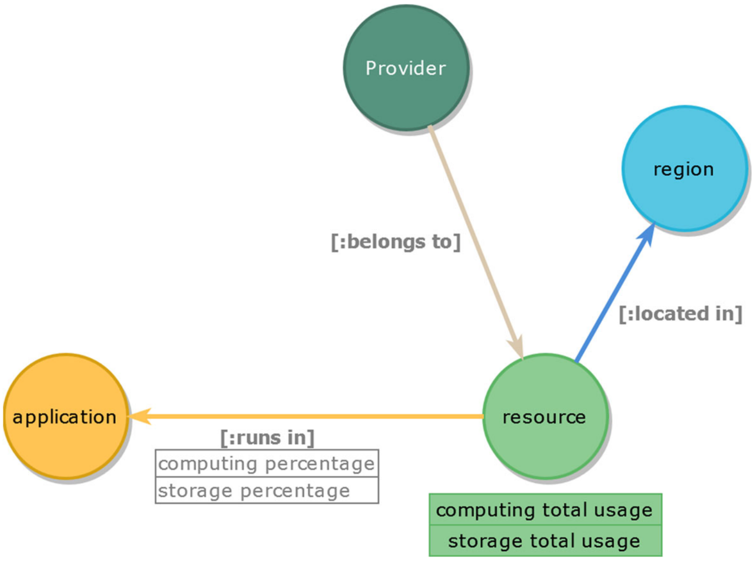


Fig. 3 Knowledge graph meta-graph

fixed-size numerical inputs (vectors). To overcome this limitation, we applied a graph embedding method to transform the graph entities (nodes, edges) into vectors of a low-dimensional space, while preserving the graph's topology. The preservation of graph topology in the embedded space ensures that the spatial relation-

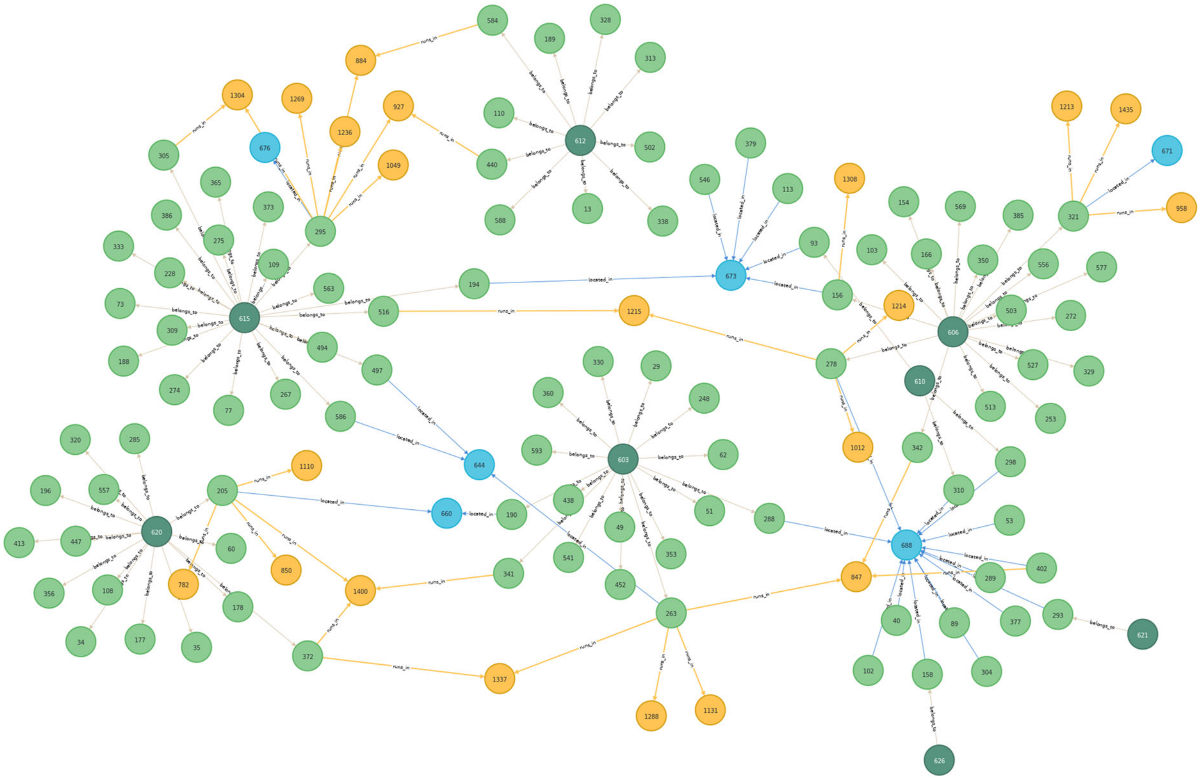


Fig. 4 Snapshot of the simulated knowledge graph

ships between the nodes, reflected by the edge structure in the original graph, are not lost in the transformation. This approach thus allows us to utilize conventional ML techniques on the resulting embedding, which is a fixed-size numerical matrix.

The embedding process is denoted as a function:

$$f : (V, E) \rightarrow \mathbb{R}^d \tag{1}$$

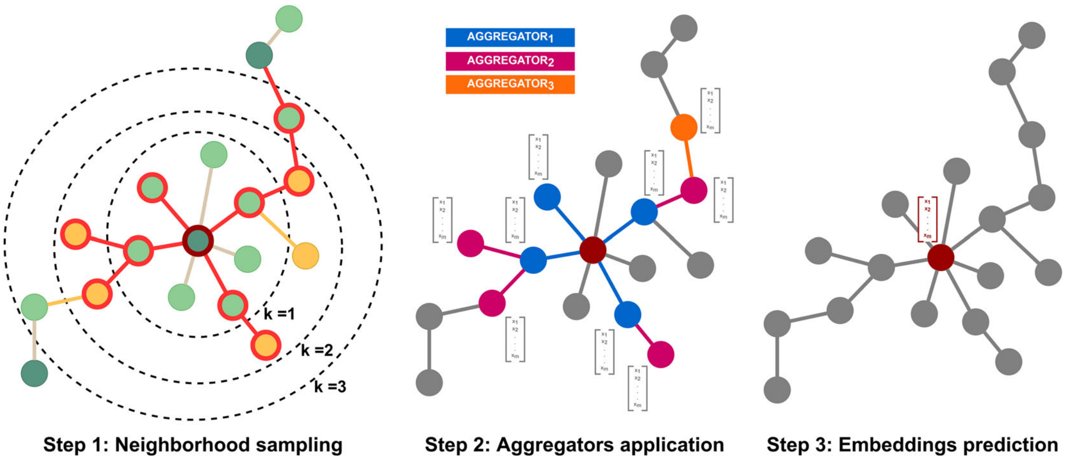
where V is the set of nodes, E is the set of edges, $\mathbb{R}^d$ denotes the real-valued vector space where the graph is embedded, with d being the specified dimension of the embeddings. The function f (1) is a graph embedding process that captures the structure and attributes of the graph in the d-dimensional space.

We used GraphSAGE [54], a neural-based, graph embedding method to train a model on a sub-graph of the aforementioned graph. The sub-graph contains nodes of type `providers`, `resources` and `applications`, as well as their in-between relationships and any properties attached to them. GraphSAGE is capable of generating predictive representations in a fully unsupervised manner by sampling and aggregating features from a node's local neighborhood using random walks. Hence, instead of training a standalone embedding vector for each node, a set of aggregator functions that combine feature information from its closest neighbours is trained. By incorporating the node features in the learning algorithm, the capability of learning the topological structure of the node's neighborhood and the distribution of the node features in it, is simultaneously introduced.

For each node $v \in V$ of the sub-graph, GraphSAGE creates a tree that has as root the corresponding node. The depth of the tree is set to the defined search depth K inside the graph, and each tree node's children are its adjacent nodes in the graph. To keep the computational footprint of each batch fixed, a fixed-size uniform sampling of the immediate neighbors is used, instead of using the full immediate neighborhood sets. In Fig. 5, Step 1 depicts the random tree of depth three (3) created for a node of the graph. These trees are then used by the aggregation functions to create embeddings of the root node of each tree.

3.2.1 Embedding Process

The graph embedding process involves an iterative method that is based on the network topology and on the features of the neighbouring nodes (Fig. 5). Initially, given a target node and a defined range of search depths K, for each $k \in \{1, \ldots, K\}$ the algorithm updates the representations of the nodes by aggregating the previous representations of its immediate neighbours (i.e. the nodes in the $(k-1)^{th}$ depth are updated based on the features of the nodes in the k^{th} depth). For this purpose, one of the aggregation mechanisms described in the following subsection are used. Next, this neighbourhood representation is concatenated with the node's previous representation and finally this concatenated vector



Step 1: Neighborhood sampling

Step 2: Aggregators application

Step 3: Embeddings prediction

Fig. 5 The three-step process of the GraphSAGE inductive representation method

is fed through a fully connected layer with a nonlinear activation function, which transforms the representation to a fixed size. Hence, as the process iterates through search depths, nodes incrementally gain more and more information from further reaches of the graph (thus each iteration can be viewed as a layer of depth influence).

For $k = 0$, the algorithm initializes by setting as representations of each node its input node features. Next, as shown in Step 2 of Fig. 5, for $k = 1$ the representation of the red node will be updated with the aggregated information derived from its blue neighbour nodes, for $k = 2$ the representation of the blue nodes will be updated by that of their pink neighbour nodes, and so on. Finally, the representation of the target node (red) is derived by the updated representations of its immediate neighbours, which are aggregated into a single vector (Step 3 of Fig. 5). This (fixed) vector representation comprises the final embeddings for the target node; this operation is repeated for every node in our graph.

3.2.2 Aggregation Mechanisms

The aggregation of the neighbor representations can be performed by a variety of aggregator mechanisms that operate over an unordered set of vectors; contrary to most datasets, a node's neighbors have no natural ordering. Therefore, the aggregation function must have the symmetry property (i.e., to be invariant to permutations of its inputs). An intuitive choice would be the mean aggregator, which is a mean operator that produces the elementwise mean of the input vectors. Other alternatives include long-short term memory (LSTM) networks and pooling aggregators [55]. LSTMs are not inherently symmetric by design, so the LSTM aggregator is based on an adapted LSTM mechanism that operates on an unordered set by applying the LSTMs to a random permutation of the node's neighbors. A pooling aggregator feeds each neighbor's vector through a fully-connected single-layer neural network, independently and an element-wise max-pooling operation is applied to aggregate information across the neighbor set. In any case, thanks to the symmetry property, a neural network-based aggregator can be trained and deployed to arbitrarily ordered feature sets.

During training, a graph-based loss function is leveraged in a fully unsupervised learning setting to tune the learnable weight matrices W^k via stochastic gradient descent, as depicted in (2). The loss function L incorporates a negative sampling term, thus encouraging nearby nodes to have similar vector representations while at the same time enforcing the representations of disparate nodes to be highly distinct. The learning rate α is a hyperparameter that controls the step size in the gradient descent update.

$$W^{(k)} := W^{(k)} - \alpha \frac{\partial L}{\partial W^{(k)}} \tag{2}$$

At inference time, the trained system is used to generate embeddings for entirely unseen nodes by applying the learned aggregation functions. GraphSAGE follows an inductive approach that only exploits local node attribute information, thus it is inherently generalizable to unseen data, in contrast to transductive embedding frameworks that can only generate embeddings from static graphs.

3.3 Event Prediction

In our experiments, we consider two types of events within the infrastructure: computing overusage and storage overusage. The computing overusage event occurs when the total computing power requested by the applications served in a node exceeds its capacity. Similarly, the storage overusage event occurs when the storage requested by the applications exceeds node's capacity. The embeddings described in the previous subsection were utilized as input for anomaly detection algorithms that were trained to identify these two types of events.

The elements of each vector were considered as features during the anomaly detection phase and thus the size of the tabular dataset used for the training of the algorithms is determined by the number of nodes in the sub-graph and the length of the vectors.

Anomaly detection algorithms fall under two main categories, distance-based and density-based. Distance-based algorithms perform better when the anomalies at hand are instances with fewer than p neighboring points within a distance D. On the contrary, density-based algorithms perform better when the anomalies are instances in low density regions or low local/relative density. Given that the embeddings created by the GraphSAGE algorithm encapsulate topology-related and feature-related information, we exploited methods from both categories to identify the events and compare the results. Specifically, we used two unsupervised methods to perform multivariate anomaly detection on the embeddings, Cluster-based Local Outlier Factor (CBLOF) [56] and Isolation Forest [57]. The former is density-based and the latter distance-based.

CBLOF calculates the outlier score using the ”cluster-based local outlier factor” measure. Local outlier factor [58] is based on the local density of a data point, where locality is given by its k nearest neighbors and density is estimated by their distance. By comparing the local density of a point to the local densities of its nearest neighbors, regions of similar density can be identified, as well as points that exhibit a substantially lower density than their neighbors. These points are considered to be outliers. Consequently, cluster-based local outlier factor is measured by both the size of the cluster to which the point belongs and by the distance between the point and its closest large cluster, if it belongs to a small one.

The algorithm consists of two high-level steps. Initially, it utilizes a clustering algorithm to partition the data in disjoint subsets; the choice is irrelevant since the only requirement is to perform well on the dataset and produce good clustering results. Subsequently, the clusters are separated into two categories, small clusters (SC) and large clusters (LC), with respect to parameters α and β. A boundary b that separates SC and LC is computed as follows: The clusters are ordered by size, and the b larger are selected so that either the proportion of number of samples in all LC is greater than α or the size ratio between the smallest LC and largest SC is greater than β. Finally, the cluster-based local outlier factor is calculated for each point based on the size of the cluster it belongs to, as well as the category it falls under. If the point belongs to a LC, the score equals to the size of the cluster multiplied by the distance of the point from it. If the point belongs to a SC, the score equals to the size of the nearest LC multiplied by the distance of the point from it. The measure used for the computation of the distance between the point and the respective cluster depends on the clustering algorithm used in the initial step; the similarity measure must be the same.

Figure 6 depicts an example of the process, in which 8 clusters have been created (C_{1-4} are LC, while C_{5-8} are SC). The centre of some clusters has been denoted for visualization purposes. As seen, for points belonging to C_1 or C_2 the distance between them and their respective cluster is calculated (yellow lines), while for points belonging to C_5 or C_8 the distance between them and the closest LC is calculated (C_3 or C_2 respectively).

Similarly to Random Forest [59], Isolation Forest is built on an ensemble of binary (isolation) trees. Their difference lies in the detection of anomalies using isolation, rather than modelling of the normal points, where

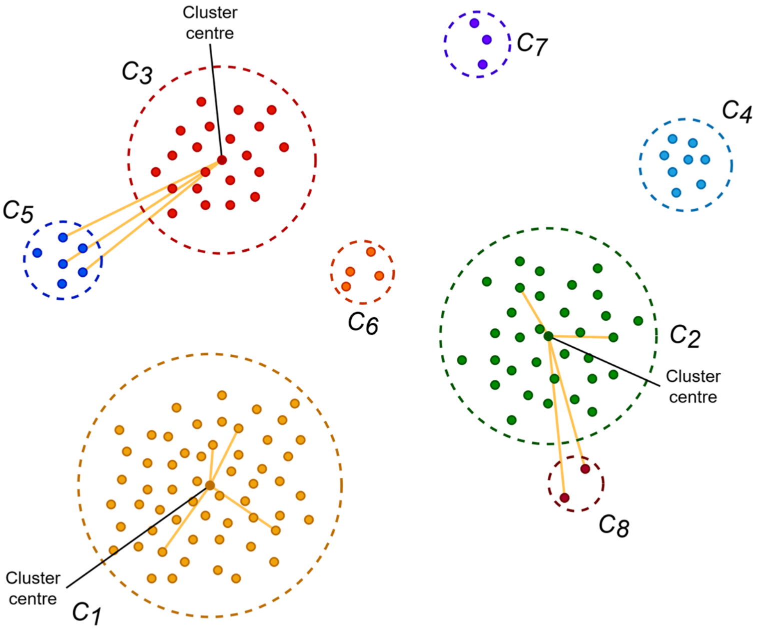


Fig. 6 Example clustering of the CBLOF algorithm

isolation is measured in terms of distance between a data point and the rest of the data. The rational is that anomalous instances (instances with distinguishable attribute-values) are more likely to be separated in early partitioning, as shown in Fig. 7. Hence, they tend to be easier to separate from the rest of the sample, compared to normal points.

In order to isolate observations, the algorithm selects a set of subsets of the original dataset (of defined size) and constructs an isolation tree per subset. Each isolation tree is created as follows: a feature is selected randomly and a random corresponding split threshold, the value of which lies between the maximum and minimum values of the selected feature. This processes is performed recursively until either the tree reaches a height limit or all observations are separated. Since recursive partitioning can be represented by a tree structure, the number of splits required to isolate a sample is equivalent to the path length from the root node to the terminating node, as depicted in Fig. 8. The average path length, produced by the set of similar random isolation trees (isolation forest), comprises a measure of normality and the decision function. Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random isolation trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies (in Fig. 8, shortest paths of each tree are denoted with darker colors).

3.4 Resource Re-Allocation

When an anomaly event is detected, a resource optimization mechanism is required to alleviate the cause of the event and to maintain optimal performance of the infrastructure and the running applications. Our solution sets up a closed loop operation between event detection and resource allocation, in which the resource allocation procedure is reactively executed based on the events received, reallocating application workload to other unaffected resources.

The event detection mechanism brings an added layer of interpretability and explainability to our model. It operates as a filter layer that isolates the subgroup of the original infrastructure affected by the anomalous behaviors and conveys this information to the re-allocation mechanisms, specifying not merely the occurrence of an anomaly but detailed information about the corresponding nodes.

To actualize this, we developed a simple but efficient heuristic algorithm (Algorithm 1) based on the Best-Fit approach, with the objective to minimize the number of re-allocations required to handle an event. The input of the proposed mechanism includes the infrastructure's topology and characteristics (e.g., resources' capacity) and information regarding the current resource allocation. Upon an anomaly event triggering its execution, the mechanism first identifies the applications affected by the overusage event. It then examines the applica-

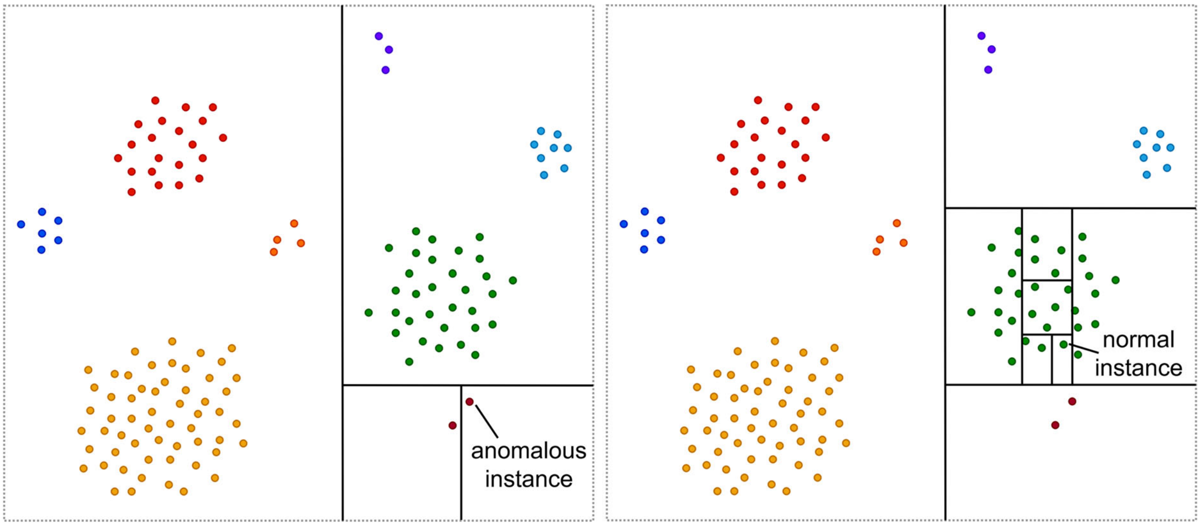


Fig. 7 Example partitioning of the Isolation Forest algorithm

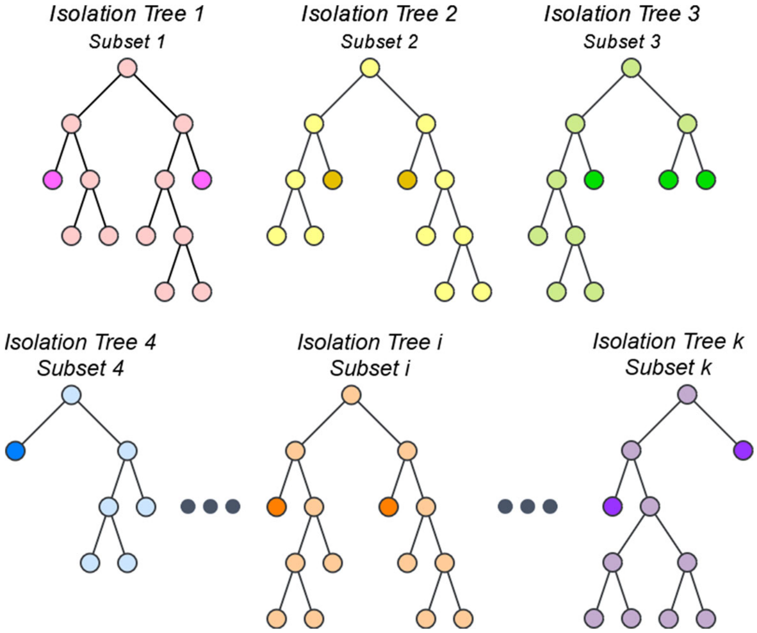


Fig. 8 Disparity between the path lengths of normal and anomalous samples

tions' demands in the respective node and calculates all possible migrations that can alleviate the overusage. These solutions are subsequently ordered based on the number of re-allocations required. Finally, each candidate solution from the ordered list is evaluated to find the resources that can serve the application's demands.

This process thereby ensures an explainable and interpretable reallocation process that is based on the detected anomalies, making it a more effective and understandable tool for managing resources within the network.

Algorithm 1 Resource optimization algorithm.

```
Input: Nodes' computing/storage utilization profiles, application assignments, anomaly events
Output: Allocation of applications to computational resources
Sort the nodes based on their utilization profiles in ascending order
for each application related to a resource node where an event was triggered do
    Calculate the number of required re-allocations
end for
Sort the applications based on the required number of re-allocations in descending order
for each candidate for re-allocation application do
    if node has enough to serve the application then
        Assign application to node
    else
        Search for resource availability among the remaining nodes
    end if
    if a node with enough capacity exists then
        Assign the application to the first one found
    else
        Blocked
    end if
    Update the nodes' resource utilization profiles
    Update applications' assignments
end for
```

Overall, our methodology significantly addresses the complexity inherent in monitoring cloud computing infrastructures, by leveraging KGs and GraphSAGE embeddings to convert intricate and high-dimensional monitoring data into a structured and manageable form. More specifically, the KG represents diverse entities and their interrelations within the cloud environment, transforming the otherwise convoluted data into an intelligible graph structure. Subsequently, GraphSAGE embeddings further distill this complexity by encapsulating both the topological and feature-related information of these entities into lower-dimensional, meaningful vector representations. This approach not only simplifies the complex monitoring process, but also enhances the effectiveness of our unsupervised machine learning algorithms in detecting anomalies and optimizing resource allocation. By reducing the dimensionality and organizing the data more coherently, our method facilitates a more efficient analysis and interpretation of the cloud infrastructure's state, thus overcoming the challenges posed by the vast and multifaceted nature of monitoring data in cloud environments.

4 Experimental Setup

We used the NetworkX Python package [60] in order to simulate an integrated computing and storage infrastructure. Neo4j Python Driver [61] was also leveraged to import the graph to the Neo4j graph database management system [62] and represent it as a knowledge graph. Overall, we consider 600 `resources` belonging to 30 different `providers` via 600 `[:belongs to]` relationships. These `resources` are allocated to 60 `regions` via 600 `[:located in]` relationships. Finally, 800 `applications` are connected with the available `resources` using 2414 `[:runs in]` relationships. Each `resource's` parameters include a *computing total usage* and a *storage total usage* property, while each `[:runs in]` relationship contains a *computing percentage* and *storage percentage* property, denoting the infrastructure needs for the corresponding application.

Neo4j's built-in GraphSAGE algorithm was used to create node embeddings. We built a 3-layer GraphSAGE architecture using a *pool* aggregation strategy, a random walk search depth of $w = 5$ and the sigmoid activation function [63]. In our initial experiments with the aggregation strategy for GraphSAGE, we explored the use of an LSTM aggregator. However, we observed underwhelming results, which can be attributed primarily to the unordered nature of our graph data. LSTM models, being inherently sequential, rely on the order of input data to capture temporal or sequential dependencies. In the context of graph data, where nodes and their connections do not possess a natural or intrinsic sequence, the LSTM aggregator struggled to effectively process and leverage this type of data structure, leading to worse anomaly detection performance. This, combined with the increased complexity and computational requirements of LSTM, informed our decision to favor the pooling aggregator for GraphSAGE. The GraphSAGE model was trained on a batch size of $b = 10$ for 30 epochs and a learning rate of $l = 0.1$ in order to extract node embeddings of dimension of $d = 16$. The trained model was used to induce embeddings for every node of the sub-graph, which were then added as additional properties of type *embeddingGraphSage*.

Finally, we used PyOD's [64] CBLOF and Isolation Forest algorithms to detect anomalies. We trained the CBLOF algorithm to take as input 16 clusters produced by the trained K-Means clustering algorithm [65]. Euclidean distance was used by K-Means and consequently by CBLOF to calculate the distance between instances in a feature array. Regarding the coefficients for determining small and large clusters, we set the alpha parameter $\alpha = 0.839$ and the beta parameter $\beta = 2$. We also set the contamination factor of the dataset to be equal to the default, $c = 0.1$.

We trained the Isolation Forest with e = 100 base estimators in the ensemble and f = 1 dataset feature for each base estimator. Sampling without replacement was performed and the number of samples drawn from the dataset to train each base estimator was set to be equal to $\min(256, s)$, where s = number of samples. We also set the contamination factor of the dataset to be equal to the default, $c = 0.1$.

4.1 Anomaly Detection Evaluation

We consider two types of events within the infrastructure: computing overusage and storage overusage. In the repetitions of our experiments, the running applications exceed the provided capacity up to 30%, creating multiple concurrent events. Each simulation scenario sets up the knowledge graph inducing the various events, creates the embeddings and then identifies the events. The evaluation metrics utilized in order to showcase the efficiency in identifying the events using the implemented methodology are Confusion Matrix, Accuracy, Precision, Recall and finally F1 Score. A brief description of each is as follows:

Confusion Matrix: Confusion matrix represents the error of a classification problem such that value C_{ij} of cell (i, j) is equal to the number of observations belonging to group i but predicted to be in group j. In a binary classification scenario, value $C_{0,0}$ (True Negatives-TN), represents the number of observations that are negative and are correctly predicted as negative. On the same note, value $C_{1,1}$ (True Positives-TP), is the count of positive observations that are correctly predicted as positive. On the other hand, value $C_{0,1}$ (False Positives-FP), is the number of observations which are originally negative but are wrongfully predicted as positive, and value $C_{1,0}$ (False Negatives-FN), is the count of originally positive observations that are wrongfully predicted as negative.

Accuracy: Accuracy is an intuitive performance measure that gives a general idea about how well a model is trained. It computes the ratio between predicted observations and total observations as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision: Precision explains what percentage of the correctly predicted cases actually turned out to be positive and is useful in the cases where False Positive is a higher concern than False Negatives. It can be calculated as:

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall explains how many of the actual positive cases were predicted correctly. Contrary to precision, recall is useful in cases where False Negative is of higher concern than False Positive. It can be calculated as:

$$Recall = \frac{TP}{TP + FN}$$

F1 Score: F1 Score gives a combined idea about Precision and Recall and is maximized when Precision is equal to Recall. It can be calculated as:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

We composed a summary comparative table (Table 1) that includes the model name, as well as the above described metrics.

From Table 1, we can easily observe that the reported accuracy of both models is satisfactory as it surpasses 97% mark. However, CBLOF outperforms Isolation Forest on the rest three metrics. This can be partially attributed to the fact that CBLOF follows a local approach, being capable of identifying outliers in subsets of the dataset (i.e., topology) that may not be considered outliers in other parts. Due to the vector representation provided by GraphSAGE, all nodes can be considered as points in a multidimensional space. As such, outliers are likely to form less densely populated areas compared to normal cases, which enables clustering-based outlier detection approaches like CBLOF to correctly identify these as potential anomalies. We also provide a confusion matrix for each model (Figs. 9 and 10) as an alternative way to evaluate their performance.

Table 1 CBLOF and Isolation Forest evaluation metrics

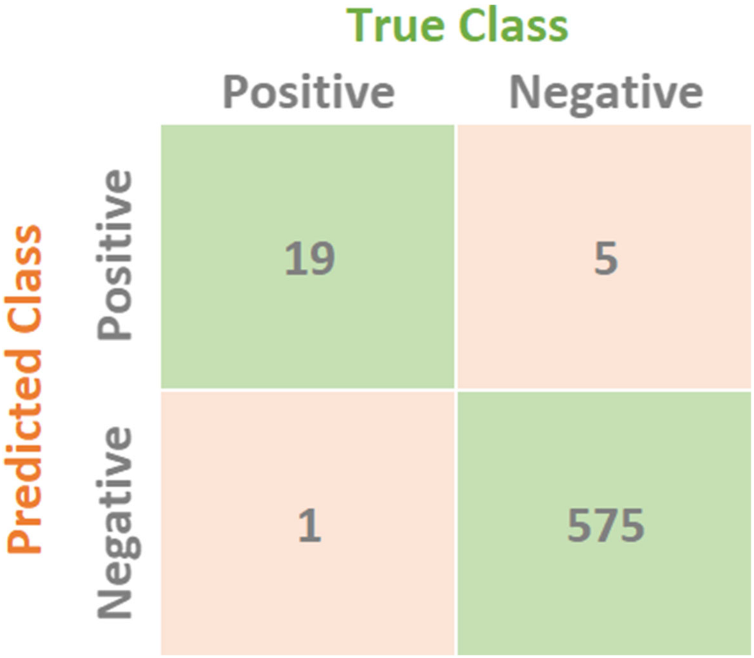
	Accuracy	Precision	Recall	F1 Score
CBLOF	0.990	0.792	0.950	0.864
Isolation Forest	0.978	0.621	0.900	0.735

The outcome of the metrics in Table 1, also depicted by the confusion matrices above, render obvious that Isolation Forest has a higher number of false positives and false negatives compared to CBLOF. We can see that although both models have similar accuracy, CBLOF has a significantly smaller number of false positives, which leads to significant difference in precision. We also provide a qualitative manner of evaluating the performance of both approaches, via the scatter plots of Figs. 11 and 12. The x-axis represents the embeddings of each simulated network node, while the y-axis represents their anomaly score (an outlier should have a higher anomaly score compared to a normal sample). It is evident that in both cases there is a clear distinction between anomalous observations (red points) and normal ones (green points); however, CBLOF appears to be more confident in data distribution, as the normal points are more concentrated and have a lower average anomaly score compared to those of Isolation Forest. This further points to the fact that the training scheme of Isolation Forest results in a more accurate approximation of the underlying data distribution.

These remarks led us to the conclusion that CBLOF seems to be more sensitive in correctly identifying the anomaly class, producing better results.

4.2 Evaluation of the Closed-Loop Control

We evaluated the performance of the resource allocation mechanism that operates in a closed-loop fashion when events are detected. We assumed that the total capacity of the infrastructure nodes is 100 computing units and 200 storage units. Application demands were generated according to a Poisson process with average computing and storage requirements in the close interval of [14,18] computing units and [20,24] stor-

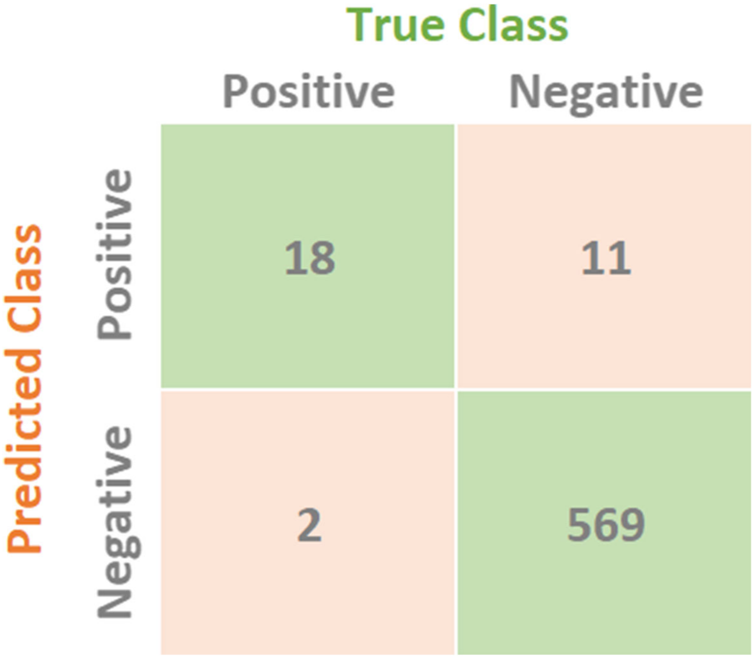


		True Class	
		Positive	Negative
Predicted Class	Positive	19	5
	Negative	1	575

Fig. 9 CBLOF confusion matrix

age units, respectively. The performance of the proposed mechanism was compared to the case where no closed-loop control was implemented, by analyzing the probability of application blocking due to insufficient resources (# of affected applications) and average resource utilization.

The results of the simulations experiments indicate that the proposed resource re-location procedure, triggered by the event-detection mechanism, reduces significantly the number of affected applications (Fig. 13) in comparison to when such mechanisms are not used. This is attributed to the improved resource utilization,



		True Class	
		Positive	Negative
Predicted Class	Positive	18	11
	Negative	2	569

Fig. 10 Isolation Forest confusion matrix

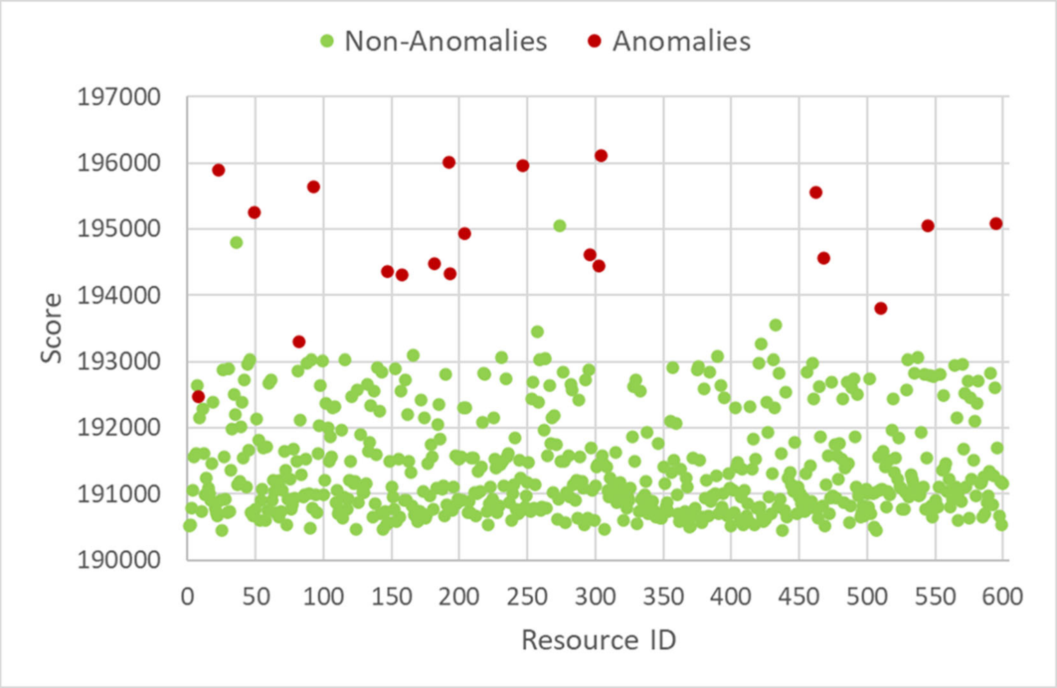


Fig. 11 Scatter plot of CBLOF. Red points represent overusage events, green points represent normal node states

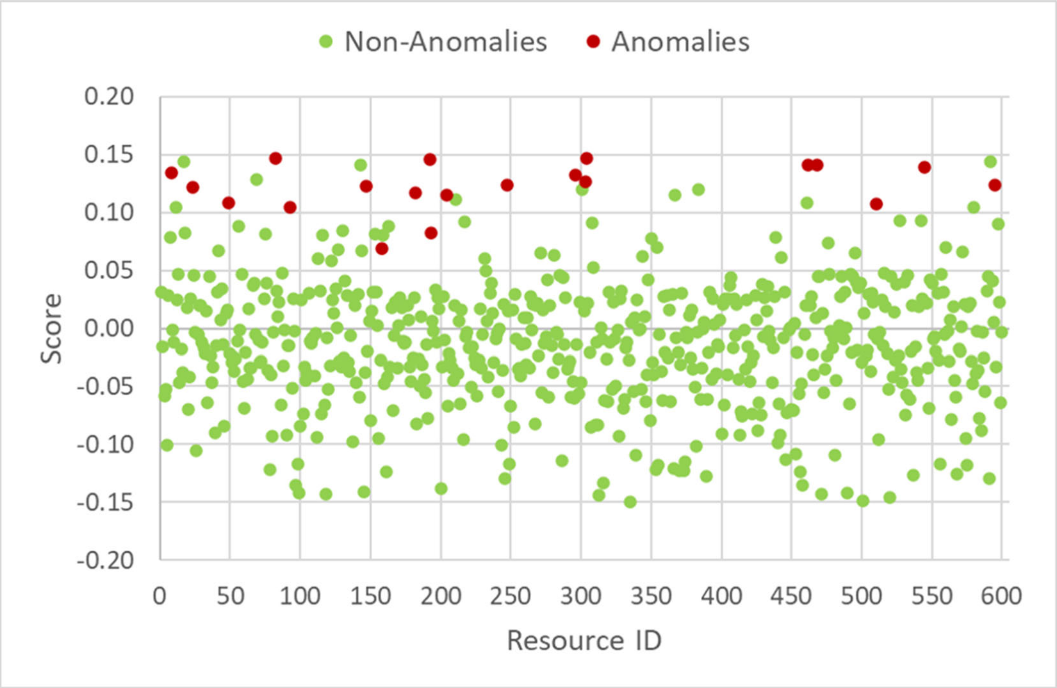


Fig. 12 Scatter plot of Isolation Forest. Red points represent overusage events, green points represent normal node states

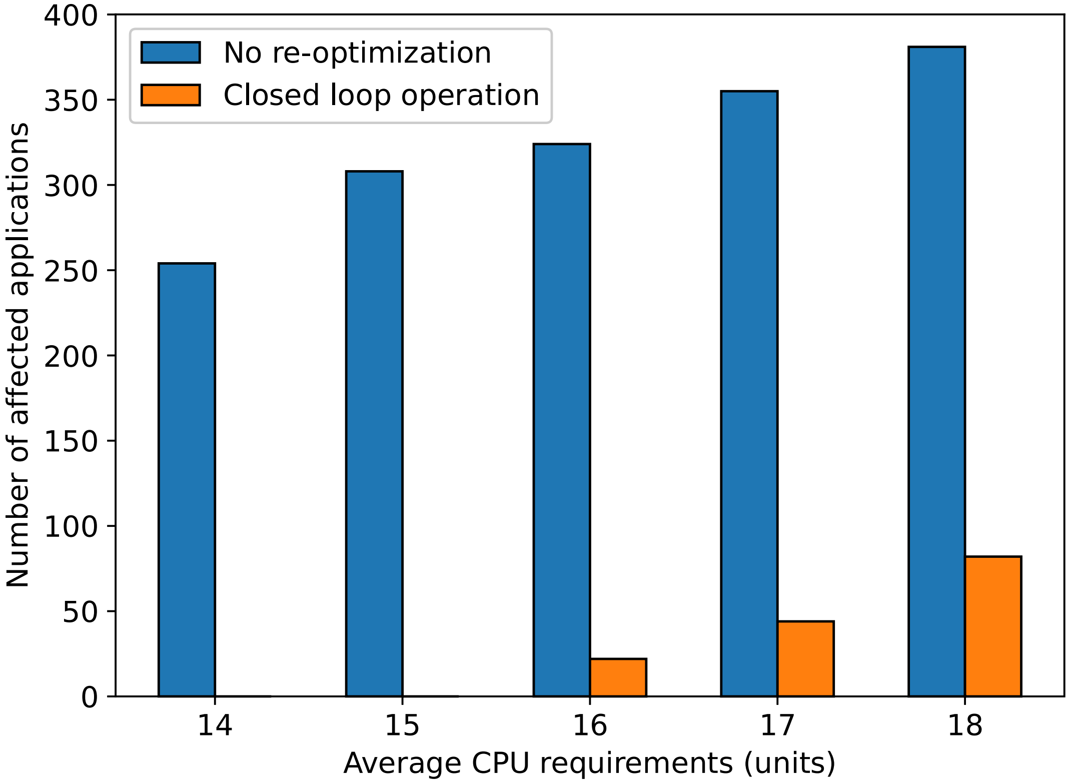


Fig. 13 Affected applications for different average applications' computing units requirements

resulting from the migration of applications' workload to other infrastructure nodes. In particular, the resource re-optimization procedure improves the utilization efficiency of the infrastructure, up to 50% higher for all the examined scenarios than when such mechanisms are not executed. This is depicted in Fig. 14 where we present the outcomes of our simulation experiments to assess the efficacy of the closed-loop resource optimization mechanism. The bars in the figure correspond to the left vertical axis and represent the number of events that triggered the re-optimization mechanism in the closed-loop operation. The lines, contrasting the bars, correspond to the right vertical axis and denote the average utilization of computing resources in the same scenarios.

5 Limitations and Threats to Validity

In the development of our methodology, we initially sought to employ real-world datasets or benchmarks to enhance the applicability and robustness of our findings. However, challenges in finding datasets that closely aligned with the specific requirements of our study led us to opt for simulated data. Below, we discuss why certain datasets, including ones from EURO28 and US26 topologies, as well as selected datasets from the Grid Workloads Archive (GWA), were not suitable for our research needs:

1. Network-Centric Datasets: Datasets based on EURO28 and US26 topologies [66] provide data on node locations, traffic volumes, and routing paths. While valuable for network behavior studies, these datasets do not encompass the broader cloud orchestration context required for our research, such as detailed information about cloud-specific dynamics like virtual machine performance, storage capacity, and real-time scalability [67].
2. Grid Workloads Archive (GWA) Datasets [68]: Several datasets from the GWA, which offer real-world workload traces from grid and cloud computing environments, were also considered. However, these datasets predominantly focus on workload patterns and resource usage in grid and cloud

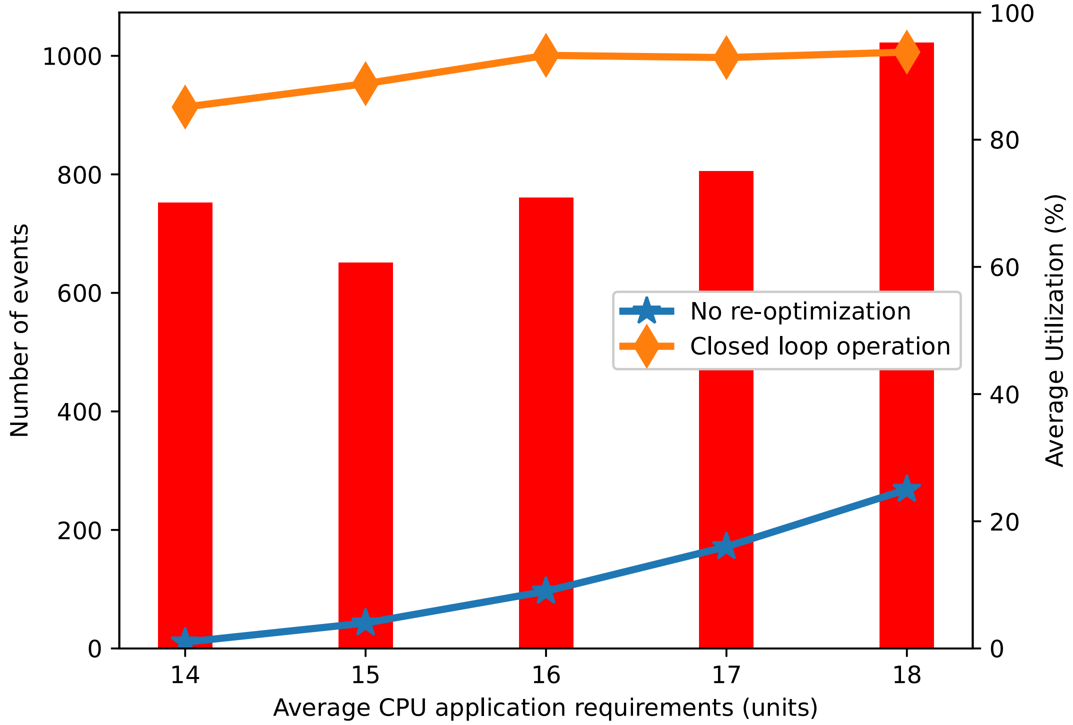


Fig. 14 Average utilization for different average application computing units requirements. Red bars denote the number of events that triggered the re-optimization mechanism in the closed-loop operation, orange points denote the average utilization in the closed-loop operation, blue points denote the average utilization in no re-optimization scenarios

environments, but lack comprehensive data on the dynamic interaction between different cloud resources and applications. Examples of such datasets include GWA-T-2 Grid5000 [69] and GWA-T-4 AuverGrid [70], which, while providing insights into workload patterns and basic resource usage, do not offer the necessary cloud resource interdependencies for our experiments. The most comprehensive of them, GWA-T-12 BitBrains [71], does provide the detailed computing and memory metrics, but still lacks wider orchestration context, such as interactions between various cloud resources and their collective response to application demands.

Given these limitations, we opted for a simulated environment tailored to our specific research needs. This approach allowed us to create scenarios that closely mimic real-world cloud computing environments. To comprehensively assess the robustness and applicability of our study, we delve into the following key aspects:

1. Generalizability of findings: Our methodology is designed to be generalizable and not limited to the specifics of the simulated data. The principles and mechanisms we developed can be applied broadly to various cloud computing contexts, ensuring the relevance and applicability of our findings beyond the constraints of the available real-world datasets. We acknowledge, however, that despite our efforts to accurately reflect real-world cloud computing environments, there may be complexities and variations in actual deployments that our simulations do not capture. The extent to which our results can be applied to different types of cloud infrastructures or to cloud systems at varying scales remains an area for further investigation.
2. Methodological limitations: Our algorithmic choices for representing our graph structure and detecting anomalies were driven by the specific nature of our data and our research objectives. However, this choice also implies certain limitations. For example, different graph embedding techniques or aggre-

gation methods might yield varied insights, particularly in handling the complexities of real-world cloud environments.

3. Internal validity: While our experimental setup was designed to minimize biases and control for variables, the inherent limitations of a simulated environment may affect the internal validity of our study. We acknowledge that our control over certain variables in a simulated environment does not perfectly replicate the unpredictability and variability of real-world settings.
4. External validity: The applicability of our findings to settings outside of the ones we simulated is not fully assured. Factors such as different cloud architectures, diverse application requirements, and varying scales of infrastructure could impact the effectiveness of our proposed methodologies. Further empirical validation in diverse real-world environments would enhance the external validity of our work.
5. Replicability and transparency : We have strived for transparency in our methodology and data simulation processes to facilitate the replicability of our study. Nevertheless, variations in computational environments and libraries and in the interpretation of our simulation parameters could impact the ease of replication. We encourage other researchers to adapt and modify our approach to suit their specific contexts and constraints.
6. Future work: Our research opens several avenues for future exploration. This includes the application of our methodology to real-world datasets, the comparison of our chosen combination of algorithms (e.g. GraphSAGE, pooling aggregator, CBLOF) with alternative approaches, and the extension of our work to encompass a wider variety of cloud computing scenarios. Continued research in these areas will help in validating and refining our approach, contributing further to the field of anomaly detection in cloud computing.

In conclusion, while our study offers valuable insights into the application of knowledge graphs and machine learning in cloud computing, we acknowledge a number of limitations regarding the generalizability and validity of our approach, highlighting the need for further research to build upon and extend our findings.

6 Discussion

In this section, we provide an overview of the added value that stems from the application of our approach in the cloud, with regard to the two main stakeholder groups; service providers and end users.

Our proposed methodology offers a new perspective for service providers, functioning as a dynamic and adaptive model of their infrastructure. It significantly deviates from the traditional rule-based approaches by proactively discovering usage patterns and predicting resource overusage events. Traditional rule-based methods adopt additive approaches that are difficult and time-consuming to sustain, while timing is crucial in the context of the fast-paced and dynamic environment of the edge and cloud infrastructures. In contrast, our approach's distinct ability, free from predefined rules and reactive measures, gives it a competitive edge as it seamlessly and rapidly adapts to the evolving conditions of the network without the need of manual intervention.

Our approach adjusts to the unique characteristics and patterns of the infrastructure, contrary to the existing rule-based methodologies that apply static procedures regardless of the network's peculiarities. This results in more accurate resource allocation predictions, subsequently leading to fewer Service Level Agreement violations (availability, response time, reliability, cost limit). Additionally, it enables streamlining the management of heterogeneous resources across multiple domains, not only by enhancing the overall resource efficiency but also by providing a scalable solution that caters to the rapid fluctuations in resource demand. This enables the implementation of complex billing models by forecasting future capacity needs with enhanced precision and accuracy, further underlining the unique benefits of our approach.

From the end-user's perspective, the user can be notified in time for potential outlier events regarding irregular resource consumption, which can be the result of compromised, malfunctioning or corrupted services, software integrity violations or configuration anomalies. It should be noted that in the aforementioned cases, while none of the individual resource usage values (computing and storage usage in our results) might surpass the maximum allocated value, malicious behavior can be inferred from the combination of the aforementioned values, as depicted by the KG-based embeddings of the involved infrastructure slice. Therefore, our approach adds an additional layer of protection against adversarial attacks or similar malicious intents that target the integrity of the virtualized services.

Finally, in response to the challenges presented by the complexity of monitoring data in cloud computing environments, our methodology encapsulates a detailed representation of the infrastructure while also enabling the efficient processing of vast amounts of monitoring data. The use of KGs allows for a structured and relational representation of data, turning complex and heterogeneous monitoring information into an interpretable and navigable format. This is particularly crucial in cloud environments, where the sheer volume and variety of data can be overwhelming. Furthermore, the application of GraphSAGE for creating node embeddings, aids in distilling the essence of the monitoring data, thereby facilitating anomaly detection and resource optimization. As the embeddings effectively summarize the key attributes and relationships within the infrastructure, they enable unsupervised machine learning algorithms to detect anomalies more efficiently. This approach reduces the cognitive load on system administrators and provides a more streamlined and manageable overview of the cloud infrastructure.

7 Conclusions

In our work, we have developed an innovative KG-based event detection methodology tailored for managing cloud computing resources. First, we implement a KG-based approach to encapsulate the properties of a computing and storage infrastructure where applications' workloads are distributed and transferred among the available resources. We then employ GraphSAGE to efficiently generate node embeddings based both on sampling and aggregating features from each node's local neighbourhood. This inductive approach enables the generalization of our methodology to evolving graphs characterized by previously unseen data. It contrasts with traditional methods by avoiding the training of individual embeddings for each node and instead focuses on sampling and aggregating features from each node's local neighborhood. The inductive nature of this methodology not only ensures efficiency but also guarantees its applicability to dynamic, ever-changing cloud infrastructures.

The core of our methodology lies in transforming graph properties into fixed-length vectors, which are then utilized in executing data-driven machine learning algorithms. These algorithms are specifically tasked with detecting instances where the computing and storage requirements of applications surpass the available capacity. In validating our approach, we employed two unsupervised anomaly detection algorithms, namely CBLOF and Isolation Forest, in an event detection scenario. Our experimental findings clearly demonstrate the superiority of CBLOF over Isolation Forest across all evaluation metrics. Furthermore, we integrate these event detection mechanisms with a resource re-optimization procedure, forming a closed-loop system. This integration significantly minimizes the impact of anomaly events on the cloud infrastructure and the quality of service provided. It ensures seamless application execution even in the face of resource overusage events. Our methodology stands out as a robust solution for the efficient orchestration of cloud infrastructures, adeptly overcoming challenges related to complexity, size, dynamicity of resources, and the sheer volume of monitoring data. Ultimately, this work lays the foundation for more intelligent and efficient cloud computing operations, heralding a new era in cloud resource management.

Acknowledgements This work is supported by the EU research project MARSAL (101017171).

Author contributions K.M, P.K and P.S wrote the main manuscript text. K.M. prepared the figures and software. E.V. provided funding and project supervision. All authors reviewed the manuscript.

Funding Information Open access funding provided by HEAL-Link Greece.

Data Availability Data will be made available on request.

Declarations

Competing interests The authors declare no competing interests.

References

1. Fazio, M., Ranjan, R., Girolami, M., Taheri, J., Dustdar, S., Villari, M.: A note on the convergence of iot, edge, and cloud computing in smart cities. IEEE Cloud Comput. **5**(5), 22–24 (2018). https://doi.org/10.1109/MCC.2018.053711663
2. Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., Shi, W.: Edge computing for autonomous driving: Opportunities and challenges. Proc. IEEE **107**(8), 1697–1716 (2019)
3. Bachhuber, C., Martinez, A.S., Pries, R., Eger, S., Steinbach, E.: Edge cloud-based augmented reality. In:2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), pp. 1–6 (2019). IEEE
4. Number of edge enabled internet of things (IoT) devices worldwide from 2020 to 2030. Statista (2022). https://www.statista.com/statistics/1259878/edge-enabled-iot-device-market-worldwide/
5. Tang, H., Li, C., Bai, J., Tang, J., Luo, Y.: Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud-edge environment. Comput. Commun. **134**, 70–82 (2019). https://doi.org/10.1016/j.comcom.2018.11.011
6. Soumplis, P., Kokkinos, P., Lagos, D., Kretsis, A., Sourlas, V., Varvarigos, E.: Network slicing and workload placement in megacities. In:2020 22nd International Conference on Transparent Optical Networks (ICTON), pp. 1–4 (2020). IEEE
7. Cisco annual internet Report - Cisco Annual Internet Report (2018-2023) White Paper. Cisco (2022). https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
8. Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A.: Edge computing: A survey. Futur. Gener. Comput. Syst. **97**, 219–235 (2019)
9. Christodoulopoulos, K., Sambo, N., Argyris, N., Giardina, P., Kanakis, G., Kretsis, A., Fresi, F., Sgambelluri, A., Bernini, G., Delezoide, C., et al.: Observe-decide-act: Experimental demonstration of a self-healing network. In:Optical Fiber Communication Conference, pp. 3–7 (2018). Optical Society of America
10. Svorobej, S., Bendechache, M., Griesinger, F., Domaschka, J.: In: Lynn, T., Mooney, J.G., Lee, B., Endo, P.T. (eds.) Orchestration from the Cloud to the Edge, pp. 61–77. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41110-7-4
11. Barika, M., Garg, S., Zomaya, A.Y., Wang, L., Moorsel, A.V., Ranjan, R.: Orchestrating big data analysis workflows in the cloud: research challenges, survey, and future directions. ACM Comput. Surv. (CSUR) **52**(5), 1–41 (2019)
12. Duc, T.L., Leiva, R.G., Casari, P., Östberg, P.-O.: Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. ACM Comput. Surv. (CSUR) **52**(5), 1–39 (2019)
13. Dong, D.: Agent-based cloud simulation model for resource management.J Cloud Comput **12**(1), 1–24 (2023)
14. Ashawa, M., Douglas, O., Osamor, J., Jackie, R.: Improving cloud efficiency through optimized resource allocation technique for load balancing using lstm machine learning algorithm. J. Cloud Comput. **11**(1), 1–17 (2022)
15. Yang, K., Ma, H., Dou, S.: Fog intelligence for network anomaly detection. IEEE Netw. **34**(2), 78–82 (2020). https://doi.org/10.1109/MNET.001.1900156
16. Ibidunmoye, O, Hernández-Rodriguez, F., Elmroth, E.: Performance anomaly detection and bottleneck identification. ACM Comput. Surv. **48**(1) (2015). https://doi.org/10.1145/2791120
17. Mitropoulou, K., Kokkinos, P., Soumplis, P., Varvarigos, E.: Detect resource related events in a cloud-edge infrastructure using knowledge graph embeddings and machine learning. In:2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), pp. 698–703 (2022). https://doi.org/10.1109/CSNDSP54353.2022.9908022
18. Sauvanaud, C., Kaâniche, M., Kanoun, K., Lazri, K., Da Silva Silvestre, G.: Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned. J. Syst. Softw. **139**, 84–106 (2018). https://doi.org/10.1016/j.jss.2018.01.039
19. Duan, S., Babu, S., Munagala, K.: Fa: A system for automating failure diagnosis. In:2009 IEEE 25th International Conference on Data Engineering, pp. 1012–1023 (2009). IEEE
20. Zhang, J., Zulkernine, M., Haque, A.: Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern. Part C (Applications and Reviews) **38**(5), 649–659 (2008)
21. Farshchi, M., Schneider, J.-G., Weber, I., Grundy, J.: Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis. In:2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE), pp. 24–34 (2015). https://doi.org/10.1109/ISSRE.2015.7381796
22. Fu, S., Liu, J., Pannu, H.: A hybrid anomaly detection framework in cloud computing using one-class and two-class support vector machines. In: Zhou, S., Zhang, S., Karypis, G. (eds.) Advanced Data Mining and Applications. Springer, Berlin, Heidelberg (2012)
23. Roumani, Y., Nwankpa, J.K.: An empirical study on predicting cloud incidents. Int. J. Inf. Manag. **47**, 131–139 (2019). https://doi.org/10.1016/j.ijinfomgt.2019.01.014
24. Liu, J., Chen, S., Zhou, Z., Wu, T.: An anomaly detection algorithm of cloud platform based on self-organizing maps. Math. Probl. Eng. **2016** (2016)
25. Xu, M.: A novel machine learning-based framework for channel bandwidth allocation and optimization in distributed computing environments. EURASIP J. Wirel. Commun. Netw. **2023**(1), 97 (2023)
26. Kompougias, O., Papadopoulos, D., Mantas, E., Litke, A., Papadakis, N., Paraschos, D., Kourtis, A., Xylouris, G.: Iot botnet detection on flow data using autoencoders. In:2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), pp. 506–511 (2021). https://doi.org/10.1109/MeditCom49071.2021.9647639

27. Cherkasova, L., Ozonat, K., Mi, N., Symons, J., Smirni, E.: Automated anomaly detection and performance modeling of enterprise applications. ACM Trans. Comput. Syst. (TOCS) **27**(3), 1–32 (2009)
28. Miyazawa, M., Hayashi, M., Stadler, R.: vnmf: Distributed fault detection using clustering approach for network function virtualization. In:2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 640–645 (2015). IEEE
29. Schmidt, F., Suri-Payer, F., Gulenko, A., Wallschläger, M., Acker, A., Kao, O.: Unsupervised anomaly event detection for vnf service monitoring using multivariate online arima. In:2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 278–283 (2018). https://doi.org/10.1109/CloudCom2018.2018.00061
30. Cotroneo, D., Natella, R., Rosiello, S.: A fault correlation approach to detect performance anomalies in virtual network function chains. In:2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), pp. 90–100 (2017). IEEE
31. Ullah, I., Lim, H.-K., Seok, Y.-J., Han, Y.-H.: Optimizing task offloading and resource allocation in edge-cloud networks: a drl approach. J. Cloud Comput. **12**(1), 112 (2023)
32. Jiang, F., Ma, R., Gao, Y., Gu, Z.: A reinforcement learning-based computing offloading and resource allocation scheme in f-ran. EURASIP J Adv Signal Process **2021**, 1–25 (2021)
33. Di Stefano, A., Di Stefano, A., Morana, G., Zito, D.: Prometheus and aiops for the orchestration of cloud-native applications in ananke. In:2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 27–32 (2021). IEEE
34. Nagasundaram, S., Bobinath, B., Shedthi, A., Rajalakshmi, K., Humnekar, T.D., et al.: Analysis of the requirement and artificial intelligence-based resource management system in cloud. In:2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), vol. 1, pp. 2516–2525 (2023). IEEE
35. Chen, X., Yang, L., Chen, Z., Min, G., Zheng, X., Rong, C.: Resource allocation with workload-time windows for cloud-based software services: a deep reinforcement learning approach. IEEE Trans. Cloud Comput (2022)
36. Zhang, J., Wang, J., Wu, J., Lu, Z., Zhang, S., Zhong, Y.: Warmops: a workload-aware resource management optimization strategy for iaas private clouds. In:2014 IEEE International Conference on Services Computing, pp. 575–582 (2014). IEEE
37. Guo, W., Tian, W., Ye, Y., Xu, L., Wu, K.: Cloud resource scheduling with deep reinforcement learning and imitation learning. IEEE Internet Things J. **8**(5), 3576–3586 (2020)
38. Hogan, A., Blomqvist, E., Cochez, M., d’Amato, C., Melo, G.d., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. Synthesis Lectures on Data, Semantics, and Knowledge **12**(2), 1–257 (2021)
39. Rotmensch, M., Halpern, Y., Tlimat, A., Horng, S., Sontag, D.: Learning a health knowledge graph from electronic medical records. Sci. Rep. **7**(1), 1–11 (2017)
40. Qian, J., Li, X.-Y., Zhang, C., Chen, L., Jung, T., Han, J.: Social network de-anonymization and privacy inference with knowledge graph model. IEEE Trans. Dependable Secure Comput **16**(4), 679–692 (2017)
41. Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., Guo, M.: Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In:Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 417–426 (2018)
42. Iannacone, M., Bohn, S., Nakamura, G., Gerth, J., Huffer, K., Bridges, R., Ferragut, E., Goodall, J.: Developing an ontology for cyber security knowledge graphs. In: Proceedings of the 10th Annual Cyber and Information Security Research Conference, pp. 1–4 (2015)
43. Tengku Asmawi, T.N., Ismail, A., Shen, J.: Cloud failure prediction based on traditional machine learning and deep learning. J. Cloud Comput. **11**(1), 47 (2022)
44. Xu, J., Xu, Z., Shi, B.: Deep reinforcement learning based resource allocation strategy in cloud-edge computing system. Front. Bioeng. Biotechnol. **10**, 908056 (2022)
45. Barshan, M., Moens, H., Latre, S., Volckaert, B., De Turck, F.: Algorithms for network-aware application component placement for cloud resource allocation. J. Commun. Netw. **19**(5), 493–508 (2017)
46. Tärneberg, W., Mehta, A., Wadbro, E., Tordsson, J., Eker, J., Kihl, M., Elmroth, E.: Dynamic application placement in the mobile cloud network. Futur. Gener. Comput. Syst. **70**, 163–177 (2017)
47. Sun, G., Liao, D., Anand, V., Zhao, D., Yu, H.: A new technique for efficient live migration of multiple virtual machines. Futur. Gener. Comput. Syst. **55**, 74–86 (2016)
48. Miyazawa, T., Kafle, V.P., Harai, H.: Reinforcement learning based dynamic resource migration for virtual networks. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 428–434 (2017). IEEE
49. Mijumbi, R., Hasija, S., Davy, S., Davy, A., Jennings, B., Boutaba, R.: Topology-aware prediction of virtual network function resource requirements. IEEE Trans. Netw. Serv. Manag. **14**(1), 106–120 (2017)
50. Eisen, M., Ribeiro, A.: Optimal wireless resource allocation with random edge graph neural networks. Ieee Trans. Signal Process. **68**, 2977–2991 (2020)
51. Li, W., Wang, H., Zhang, X., Li, D., Yan, L., Fan, Q., Jiang, Y., Yao, R.: Security service function chain based on graph neural network. Information **13**(2), 78 (2022)
52. Robinson, I., Webber, J., Eifrem, E.: Graph Databases: New Opportunities for Connected Data, USA (2015)
53. Cypher query language - developer guides (2023). https://neo4j.com/developer/cypher/
54. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Adv Neural Inf Process Syst **30** (2017)
55. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735. https://direct.mit.edu/neco/article-articlepdf/9/8/1735/813796/neco.1997.9.8.1735.pdf
56. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recogn. **24**(9–10), 1641–1650 (2003)
57. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Proceedings - IEEE International Conference on Data Mining, ICDM (2008). https://doi.org/10.1109/ICDM.2008.17

58. Breunig, M., Kriegel, H.-P., Ng, R., Sander, J.: Lof: Identifying density-based local outliers., vol. 29, pp. 93–104 (2000). https://doi.org/10.1145/342009.335388
59. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
60. NetworkX documentation (2023). https://networkx.org
61. Neo4j Python Driver documentation (2023). https://neo4j.com/docs/api/python-driver/current/
62. Neo4j documentation (2023). https://neo4j.com/
63. Narayan, S.: The generalized sigmoid activation function: Competitive supervised learning. Inf. Sci. **99**(1–2), 69–82 (1997). https://doi.org/10.1016/S0020-0255(96)00200-9
64. PyOD documentation (2023). https://pyod.readthedocs.io/en/latest/
65. Lloyd, S.: Least squares quantization in pcm. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
66. Wojciechowski, S., Goścień, R., Ksieniewicz, P., Walkowiak, K.: Hybrid regression model for link dimensioning in spectrally-spatially flexible optical networks. IEEE Access **10**, 53810–53821 (2022). https://doi.org/10.1109/ACCESS.2022.3175193
67. Ashawa, M., Douglas, O., Osamor, J., Jackie, R.: Improving cloud efficiency through optimized resource allocation technique for load balancing using lstm machine learning algorithm. J. Cloud Comput. **11** (2022) https://doi.org/10.1186/s13677-022-00362-x
68. Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D.H.J.: The grid workloads archive. Futur. Gener. Comput. Syst. **24**(7), 672–686 (2008). https://doi.org/10.1016/j.future.2008.02.003
69. GWA-T-2 Grid5000 Dataset (2023). http://gwa.ewi.tudelft.nl/datasets/gwa-t-2-grid5000. Accessed November 2023
70. GWA-T-4 AuverGrid Dataset (2023). http://gwa.ewi.tudelft.nl/datasets/gwa-t-4-auvergrid. Accessed November 2023
71. GWA-T-12 Bitbrains Dataset (2023) http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains. Accessed November 2023