Reinforcement Learning Multi-Agents System for Faults Diagnosis of Mircoservices in Industrial Settings

Asma Belhadi[a], Youcef Djenouri[b], Gautam Srivastava[c,e], Jerry Chun-Wei Lin*[d]

[a]*Department Department of Technology, Kristiania University College, Oslo, Norway.*
[b]*Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway.*
[c]*Department of Mathematics and Computer Science, Brandon University, Canada*
[d]*Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway.*
[e]*Research Centre for Interneural Computing, China Medical University, Taiwan*

Abstract

This paper develops a new framework called MASAD (Multi-AgentsSystem for Anomaly Detection), a hybrid combination of reinforcement learning, and a multi-agents system to identify abnormal behaviors of microservices in industrial environment settings. A multi-agent system is implemented using reinforcement learning, where each agent learns from the given microservice. Intelligent communication among the different agents is then established to enhance the learning of each agent by considering the experience of the agents of the other microservices of the system. The above setting not only allows to identify local anomalies but global ones from the whole microservices architecture. To show the effectiveness of the framework as proposed, we have gone through a thorough experimental analysis on two microservice architectures (NETFLIX, and LAMP). Results showed that our proposed framework can understand the behavior of the microservices, and accurately simulate the different interactions in the microservices. Besides, the approach outperforms the baseline methods in identifying both the local and global outliers.

Keywords: Reinforcement Learning; Multi-Agents System; Microservices;

*Corresponding author
Email addresses: asma.belhadi@kristiania.no (Asma Belhadi), youcef.djenouri@sintef.no (Youcef Djenouri), srivastavag@brandonu.ca (Gautam Srivastava), jerrylin@ieee.org (Jerry Chun-Wei Lin*)

Local Outliers; Global Outliers.

1. Introduction

Industrial applications are moving towards microservices computing. This drives to companies immediately carrying industrial settings [1]. In particular with the emergence of the IoT (Internet of Things), which plays an important role in addressing challenges of different industrial applications [**?** 2]. IIoT (Industrial Internet of Things) fosters new smart devices and applications as never seen before. Industry 4.0, medical monitorization, smart agriculture, and building are few examples of the huge potential number of IIoT applications will offer to our society. Smart sensors offered by IIoT technologies has resulted in the creation of large volumes of data varied in time and space.

Microservices is an architecture, divided into small components, each of which is a microservice that delivers a small service in the company. Companies such as Amazon, Twitter, and Netflix switched to a microservices architecture to be self-organizing and cross-functional companies [3]. Microservices in IIoT has recently shown a great interest in the data science community, where the service from the distributed and heterogeneous data is assured [4]. A useful way of analyzing microservices is by utilizing data mining and machine learning techniques [5, 6, 7]. Detecting anomalies from microservices architecture can be noted as a hot research area in IIoT. The goal is so anomalous patterns can be properly identified through data generated in the microservices architecture. However, solutions to microservices anomaly detection [8, 9, 10] are limited to identify local anomalous behavior of each microservice, where global anomalous are missing. Also, these solutions tend to lack accuracy, since there is no intelligent mechanism that can be used in any distributed analysis process.

Multi-agents systems with reinforcement learning have recently shown promising application prospects and attracted lots of attention from academia and industry [11], where it provides an efficient mechanism for interaction and communication with the different actors in the environment and to learn from the previous experiences to achieve better performances. Many of the works mentioned created reinforcement learning through a multi-agent system for commercial building [12]. Other works adopted a multi-agents based reinforcement learning approach for autonomous driving [13].

This paper follows the state-of-the-art multi-agents system reinforcement learning models, and develop a new framework to identify both local and global anomalous from the microservices architecture. The main contributions of this paper are listed as follows,

1. We propose a new multi-agent system for detecting both local and global anomalous behavior in a distributed and heterogeneous microservices architecture.
2. We use reinforcement learning to identify the local anomalous behavior in each microservice of the whole architecture.
3. We propose a new intelligent strategy to enhance the communication among the different agents, and then merge the local anomalous behavior into global anomalous ones.
4. We analyze the proposed framework on two industrial data, including NETFLIX, and LAMP. The results reveal the usefulness of our framework compared to the baseline outlier detection solutions.

The rest of this paper is organized as follows: Related work is summarized in Section 2. The proposed framework and designed algorithm are discussed in Section 3. We report our experimental results in Section 4. Section 5 concludes the paper.

2. Related Work

He *et al.* [8] proposed anomaly detection system in clouds environment using microservices. A master-worker architecture is developed, where the master requests a service to each worker to train the anomaly detection model based on the graph neural network [14]. Labiadh *et al.* [9] proposed a microservice system for identifying energy anomalous patterns by exploring knowledge transfer. It is based on the correlation between the historical energy time series data and the unseen target data. The microservice is composed of several REST APIs, one for training data selection, one for predictive model learning, one for building data handler, and the last one is for weather data handler. Jin *et al.* [10] proposed the robust principal component analysis algorithm [15] to identify outliers for microservices architecture. The anomalous score of each node is calculated using the invocation chain anomaly analysis algorithm. It then identifies the anomalous indicators of each node by combining various single anomaly detection algorithms.

Wang *et al.* [16] proposed an ensemble learning approach for capturing outliers in the microservice environment. The approach used a support vector machine and a convolution neural network in each node. Theo *et al.* [17] present an end-to-end solution for data analytic in microservice architectures. It addressed important requirements and challenges of analytic of microservices, with illustration on smart homes [18]. It also provides efficient tools such as spark to deal with big data-related problems. Berta *et al.* [19] investigated the use of data management technologies including anomaly detection in improving the microservice architectures. It can support more accurate development for different IIoT applications. The framework is applied to two relevant industrial settings such as smart homes and autonomous driving. Meng *et al.* [20] proposed a new sequential pattern mining based solution for identifying fault diagnosis. The system calls in a microservice architecture are first retrieved to build the transaction database. The sequential patterns are then discovered. The deep neural network is finally employed to model the patterns of system call sequences to diagnose faults by determining the score between the estimated next system call and the actual next one in the specific pattern. Chen *et al.* [21] introduced an unsupervised anomaly detection solution using an intelligent operator called matrix sketch. It can identify anomalies by mining high-dimensional data collected from a microservice architecture in real-time. Cui *et al.* [22] developed an optimization approach to enhance the detection of abnormal behaviors in the microservice architecture. A margin synthetic minority oversampling strategy is first performed in the imbalanced data to ensure efficient data distribution. A recursive feature elimination-hierarchy strategy is then performed to remove redundant samples recursively based on feature weight similarity. A flexible grid search algorithm is finally implemented for efficient selection of the different hyper-parameters of the learning model. In the context of intelligent agents, several solutions have been developed for microservice architectures. Rem et al. [23] considered each agent as a microservice. It developed a template of the multi-agent system for handling microservices. The proposed template can be viewed as an uniform interface for addressing industrial challenges. Petar et al. [24] developed an autonomous agents for service management in IoT settings. The microservices are considered as modern agents that might increase the systems in collaborative environment. Abeer et al. [25] proposed an autonomous agents for microservices autoscaling with quality of services conditions. It is composed of two steps, the first step applied the Kubernetes autoscaling algorithm to derive the microservice resource demand.

The second step used the intelligent agents with the reinforcement learning to learn the autoscaling threshold. Arzo et al. [26] suggested a new mutli-agent system for fully-automated network management. It developed a new network function which described the atomic decision-making parts of the network. These parts identified the virtual network, which are autonomous and adaptive.

As can be seen from the above short literature overview, existing multi agent frameworks for microservices architectures do not consider the anomaly detection process. In addition, existing solutions for anomaly detection from microservice architectures only consider local anomalies, for example, anomalies from each microservice. Discovering global anomalies from the whole architecture is vital, and need a distributed environment highly correlated. Therefore, in this paper, we propose the first framework to identify both local and global outliers from microservice architectures using reinforcement learning and multi-agent systems.

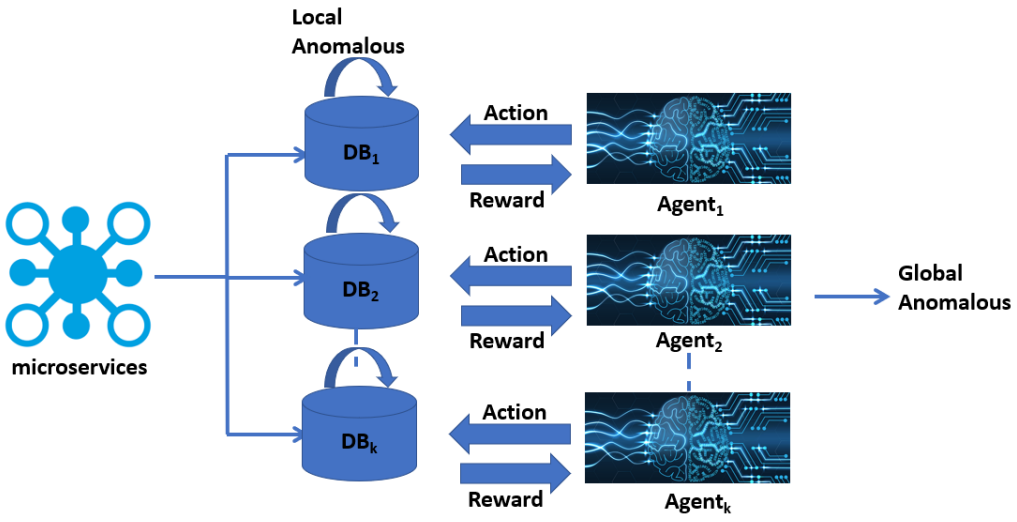

Figure 1: MASAD Framework

3. MASAD: Multi-Agents System for Anomaly Detection

Let us begin by describing the key elements of the MASAD (Multi-Agents System for Anomaly Detection). As shown in Figure 1, our framework builds

upon the multi-agents system and reinforcement learning. The framework considers microservices, each of which is requested one partition of the data. Each partition is stored in the small database. We use the intelligent agent based on reinforcement learning to identify the local anomalous behaviors from each microservice. We also use the multi-agent system for enhancing the learning process of each agent and merging the local anomalous retrieved at each agent on global ones. The whole process is ensured by an intelligent communication strategy among the different agents. MASAD can be divided into main steps:

1. **Local Outliers Determination:** The local outliers on each microservice is determined using the reinforcement learning on each agent. Each agent learns from the microservices data, by action/reward strategy (see Section 3.2 for more details).

2. **Global Outliers Determination:** After predicting the local outliers by each agent, a merging strategy is used to derive the global outliers by employing an intelligent communication strategy. Thus, the local outliers with high density are considered as global outliers (see Section 3.3 for more details).

In the remainder of this section, we describe the details of MASAD components.

3.1. Preprocessing

This step aims to preprocess the data which will be used in the next step. It is performed in two stages: The first one is the IIoT data collection from the microservices architecture to prepare it for the multi-agents system. The data is collected using the IIoT gateway, and the data is cleaned using the Apache Kafka software [27]. The second one is data enrichment by integrating ontologies to create semantic data and then make it understandable for different IIoT applications. We used EPOST (Entire Process Ontology on Software Testing) [28] to create and manipulate our ontology.

3.2. Reinforcement Learning for Local Outliers

The aim of this part is to identify the local outliers for each microservice. Each agent is assigned to one mciroservice for deriving the local outliers. To enhance the intelligent behavior of the agents, the reinforcement learning

is integrated on the inference part of each agent. We define a multi-agents system by a tuple $< \mathcal{A}, \mathcal{S}, \mathcal{U}, \mathcal{R} >$. $\mathcal{A}$ is the set of agents, each of which is a Markov decision process, $\mathcal{S}$ is the finite set of environment states, $\mathcal{U}$ is the set of actions and $\mathcal{R}$ is the reward function. The behavior of each agent in $\mathcal{A}$ is represented by its policy, which specifies how the agent chooses its actions given the state. The purpose of each agent is to find a policy that maximizes the coverage of local outliers. In the following the description of reinforcement learning concepts in retrieving the local outliers is explained:

1. **State**: The next action of each agent is dependent on the decisions of the previous states. Therefore, the state of each agent is composed of two parts, the set of the previous actions (the set of previous observations with their outlier scores), and the current data to be handled. The size of the state space $\mathcal{S}$ is measured by the number of observations in the database.
2. **Action**: It is the assignment of the anomaly decision (normal or abnormal) behavior of each observation in the database.
3. **Reward**: It is crucial to determine an appropriate reward function. It allows a better learning process of each agent in $\mathcal{A}$. We used data with ground-truth to make a reward for the actions of the agent. The reward function is defined as follows:

$$\mathcal{R}(\mathcal{A}_i, \mathcal{U}_i) = \begin{cases} 1, & \text{if } \mathcal{A}_i(\mathcal{U}_j, O_j) = \mathcal{L}(O_j); \\ 0, & otherwise, \end{cases} \tag{1}$$

   where $\mathcal{A}_i(\mathcal{U}_j, O_j)$ is the decision of the agent $\mathcal{A}_i$, whether the observation O_j is an outlier or not, and $\mathcal{L}(O_j)$ is the label of the observation O_j (outlier or not).
4. **Environment**: The environment is a set of databases of the microservices which contains a large population of microservices data. This allows the environment to generate particular states for training the agent and estimate the best actions to be taken.

Each agent $\mathcal{A}_i$ starts by scanning the observations of the i^{th} microservice, it then computes the outlier score represented by the euclidean distance between the first observation, and the remaining observations of the i^{th} microservice. If the outlier score is greater than a given threshold, an action indicating that the first observation is an outlier, otherwise, an action indicating that the first observation is normal. A reward function is computed

for this decision based on the label of the first observation. This process is repeated for all observations of the i^{th} microservice. As result, a set of local outliers noted LO_i is extracted for each agent $\mathcal{A}_i$.

3.3. Merging Strategy for Global Outliers

Our goal of this step is to learn the global outliers from the set of local anomalies of each agent. We define the global outlier pattern candidate by the set of local outliers, where each local outlier belongs to a given agent in $\mathcal{A}$. An anomalous pattern is called a global outlier if its density is greater than a minimum threshold. The density of the pattern P_i is determined by the sum of distances between each pair of elements in P_i. For normalization, we divide this value by the number of distances performed which is set to $|P_i|^2$. Formally, it is given as follows:

$$Density(P_i) = \frac{\sum_{j=1}^{|P_i|} \sum_{l=1}^{|P_i|} D(P_i^j, P_i^l)}{|P_i|^2} \tag{2}$$

Note that P_i^j, and P_i^l are local outliers of the agent $\mathcal{A}_j$, $\mathcal{A}_l$, respectively.

Our idea is based on intelligent communication among the agents in $\mathcal{A}$. Thus, the k nearest neighbors of each agent is determined based on the similarity between each two pair of agents. The similarity between the agent $\mathcal{A}_i$, and the agent $\mathcal{A}_j$ is determined as follows:

$$Sim(\mathcal{A}_i, \mathcal{A}_j) = \sum_{l=1}^{|\mathcal{A}_i|} \sum_{m=1}^{|\mathcal{A}_j|} Distance(LO_i^l, LO_j^m), \tag{3}$$

where $Distance(LO_i^l, LO_j^m)$ is the euclidean distance between the l^{th} local outlier of the agent $\mathcal{A}_i$, and m^{th} local outlier of the agent $\mathcal{A}_j$.

The process starts by computing the kNN (k Nearest Neighbors) [29] of each agent, which results $|\mathcal{A}|$ kNN sets. The global outlier pattern candidate is generated from each kNN set by taking one local outlier from each agent located in this kNN set. The density of this pattern is determined and added to the set of the global outliers if its density is greater than the minimum threshold. A greedy search algorithm is used to explore the global outlier pattern candidate space.

Algorithm 1 MASAD Algorithm

Input:
$M = \{M_1, M_2..., M_n\}$: The set of n microservices
Output:
LO: The set of local outliers.
GO: The set of global outliers.
$\mathcal{LO} \leftarrow \emptyset$.
for i=1 to m **do**
  $\mathcal{A}_i \leftarrow RL(M_i)$
  $LO \leftarrow LO \cup LO_i$
end for
$GO \leftarrow \emptyset$
$KNN \leftarrow kNN(\mathcal{A})$
for i=1 to m **do**
  $GO \leftarrow GO \cup GreedySearch(KNN_i)$
end for
return (LO, GO)

Algorithm 1 presents the pseudo-code of the MASAD algorithm. The algorithm starts by determining the set of local outliers of each microservice M_i using the reinforcement learning and by the agent $\mathcal{A}_i$. An intelligent communication among agents based on the kNN is performed to determine the global outlier pattern candidate space. This space is explored by the greedy search algorithm to identify global outliers.

The complexity cost of MASAD is the sum of the complexity cost of determining the local outliers, and the complexity cost of determining the global outliers. The complexity cost of determining the local outliers is the m times the complexity of the reinforcement learning is $O(|\mathcal{S}| \times |\mathcal{U}|)$. The complexity cost of determining global outliers is $O(|LO| \times k)$. Therefore, the complexity cost of MASAD is $O(m \times |\mathcal{S}| \times |\mathcal{U}| + |LO| \times k)$.

4. Performance Evaluation

4.1. Experimental Settings

Here, we evaluate the MASAD framework as proposed and all of the different components within it. Specifically, the framework's ability to identify local, and global anomalous patterns is analyzed using two microservice,

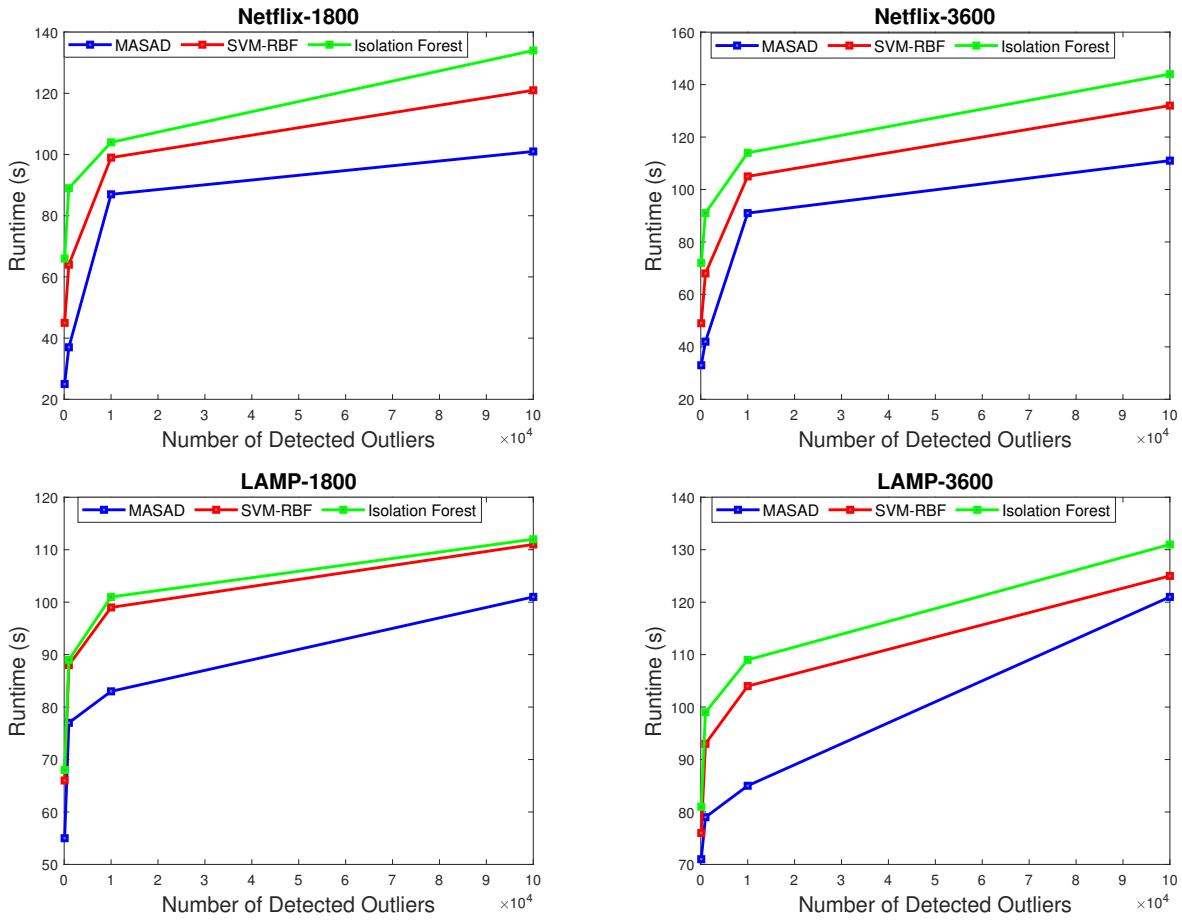

Figure 2: Runtime comparison of the MASAD and the state-of-the-art anomaly detection solutions

the NETFLIX[1], and the LAMP[2] architectures. Four datasets are generated: NETFLIX-1800, NETFLIX-3600, LAMP-1800, and LAMP-3600. Each dataset is produced in 1800 and 3600 seconds simulation on the microservice architecture NETFLIX, and LAMP, respectively.

The experimental evaluation of the implementation was undertaken on a

[1]https://netflix.github.io/

[2]https://aws.amazon.com/fr/blogs/compute/introducing-the-new-serverless-lamp-stack/

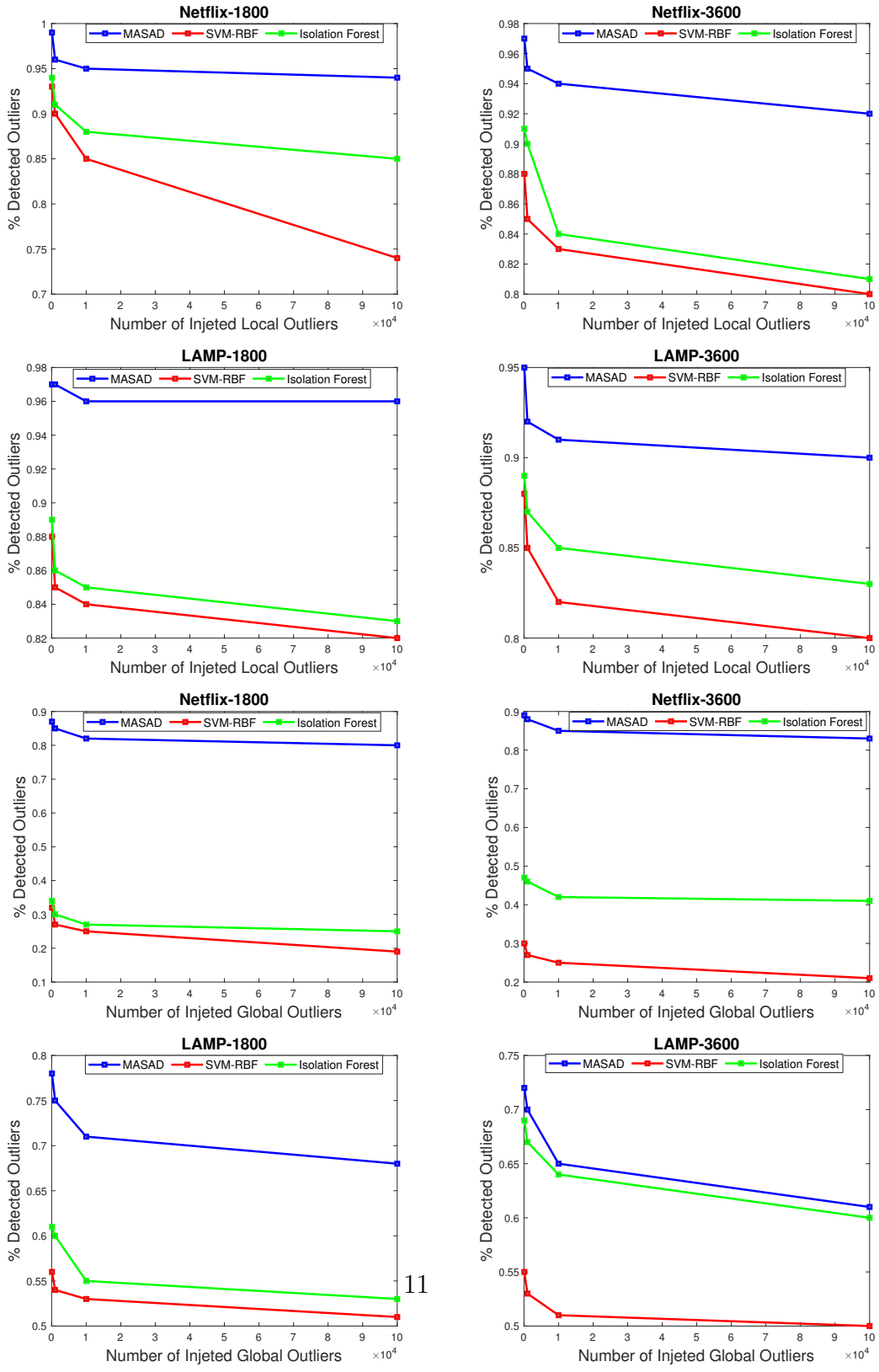

Figure 3: Comparison in terms of local outliers, and global outliers returned by MASAD and the state-of-the-art anomaly detection solutions

INTEL i7-core 64-bit processor running UBUNTU 20 and 32 GB of RAM. The host CPU is specified as a quad-core INTEL Xeon E5620 64-bit with clock measured at 3.27 GHz. The GPU is a NVIDIA Tesla C2085 with 468 CUDA cores (16 multiprocessors each with 64 cores) and clock speed of 3.15 GHz. The GPU contains 3.8 GB global memory, 59.15 KB shared memory, and warp size of 64. Single precision is used in both CPU as well as GPU. In the implementation scenario as used, GPU blocks are used for the simulation process of the multi-agents system environment. Each agent is allocated to one GPU block, where a shared memory of each block is allocated to the corresponding agent. We simulate communication among agents using global as well as constant memories of GPU host.

Generally, a well-known issue in anomaly detection is in the evaluation procedure specifically. Especially with new applications such as in IIoT applications, where real-world simulation scenarios maybe unknown. To facilitate a trust-worthy quantitative evaluation, we use the processes defined by Zhang *et al.* [30] to inject synthetic anomalous patterns.

- **Injecting local outliers**: local outliers are generated by adding noise *several times* with a certain probability $p \sim \mathcal{U}(0.8, 1.0)$ and a given threshold μ;

- **Injecting global outliers**: From the local outliers, we again add noise but now only a *few times* with a certain probability $p \sim \mathcal{U}(0.0, 1.0)$ and a given μ.

For both injections, each data d_i in each dataset is changed as follows:

$$d_i = \begin{cases} d_i + n \sim \mathcal{N}(0,1) & \text{if } d \geq \mu \\ d_i & \text{otherwise.} \end{cases} \tag{4}$$

The evaluation is performed using the ratio between the number of corrected returned outliers over the number of all outliers. This value is ranged between 0, and 1, where a higher value represents the best accuracy.

We compared MASAD against two popular solutions for identifying anomalies on large scale data. SVM-RBF (Support Vector Machine using a Radial Basis Features) [31]. Isolation Forest [32] is also considered as a baseline algorithm due to its ability in accurately retrieving outliers.

4.2. Runtime Comparison of the MASAD and the state-of-the-art anomaly detection solutions

Figure 2 presents the runtime in seconds of MASAD and the baseline anomaly detection algorithms (SVM-RBF, and Isolation Forest). Thus, several tests have been performed by varying the number of detected outliers from 100 to 100,000. Whatever the sample used as input, MASAD outperforms the two other baseline solutions in terms of computational time. In particular, for LAMP-1800 data, where the gap between MASAD and other solutions is high. This comes from the fact that the MASAD can identify anomalies more quickly using reinforcement learning. Moreover, the communication among the intelligent agents allows to rapidly identify the outliers. Contrary to the other baseline approaches, where SVM-RBF attempt to find a function to distinguish normal behaviors from others. Besides, the isolation forest algorithm creates the enumeration tree to determine the outliers.

4.3. Comparison in terms of local outliers returned by MASAD and the state-of-the-art anomaly detection solutions

Figure 3 presents the percentage of local detected outliers of MASAD and the baseline anomaly detection algorithms (SVM-RBF, and Isolation Forest). Thus, several tests have been performed by varying the number of injected local outliers from 100 to 100,000. Whatever the sample used as input, MASAD outperforms the two other baseline solutions in terms of detected local outliers. In particular, for LAMP-1800, and LAMP-3600 data, where the gap between MASAD and other solutions is high. This comes from the fact that the MASAD can identify local anomalies more quickly thanks to the learning strategy, where each agent learns from each microservice using reinforcement learning. Instead of the two other baseline algorithms where they used traditional learning approaches in retrieving the local outliers.

4.4. Comparison in terms of global outliers returned by MASAD and the state-of-the-art anomaly detection solutions

Figure 3 presents the percentage of global detected outliers of MASAD and the baseline anomaly detection algorithms (SVM-RBF, and Isolation Forest). Thus, several tests have been performed by varying the number of injected global outliers from 100 to 100,000. Whatever the sample used as input, MASAD highly outperforms the two other baseline solutions in terms of detected global outliers. This is explained by the ability of the MASAD to detect global outliers. The strategy used in the communication among

the different agents allows to share knowledge obtained by the different microservices in the system, and therefore identify global anomalies of the whole architecture. This is not the case for the traditional approaches where only local anomalies are derived.

5. Conclusion

This paper introduced a novel framework based on reinforcement learning and multi-agent system to derive both the local and the global anomalies from the microservices architecture. Results on two well-known microservice architectures showed that our proposed framework outperforms the baseline outlier detection solutions, and able to derive both the local and global outliers. Porting pure data mining, and deep learning techniques into a specific application domain requires methodological refinement and adaptation [33, 34]. In our specific context, this adaptation is implemented by integrating a new model which able to identify both local, and global anomalies from microservices data. As future perspective, advanced techniques, including recurrent auto-encoder-based approaches, should be investigated for determining anomalies in microservice architectures, in particular for determining the global anomalies. Another perspective is to handle with large microservices using the high performance computing. Exploring other type of microservices such as GraphQL is also in our future agenda.

Declaration of competing interest The authors declared that there is no conflict of interest in this study.

References

[1] M. Sollfrank, F. Loch, S. Denteneer, B. Vogel-Heuser, Evaluating docker for lightweight virtualization of distributed and time-sensitive applications in industrial automation, IEEE Transactions on Industrial Informatics 17 (5) (2020) 3566–3576.

[2] T. Wang, W. Zhang, J. Xu, Z. Gu, Workflow-aware automatic fault diagnosis for microservice-based applications with statistics, IEEE Transactions on Network and Service Management 17 (4) (2020) 2350–2363.

[3] P. Valderas, V. Torres, V. Pelechano, A microservice composition approach based on the choreography of bpmn fragments, Information and Software Technology 127 (2020) 106370.

[4] K. Bozan, K. Lyytinen, G. M. Rose, How to transition incrementally to microservice architecture, Communications of the ACM 64 (1) (2020) 79–85.

[5] Z. Wang, X. He, L. Liu, Z. Tu, H. Xu, Survey on requirement-driven microservice system evolution, in: IEEE International Conference on Services Computing, 2020, pp. 186–193.

[6] L. Chen, Y. Xu, Z. Lu, J. Wu, K. Gai, P. C. Hung, M. Qiu, Iot microservice deployment in edge-cloud hybrid environment using reinforcement learning, IEEE Internet of Things Journal.

[7] R. Wang, M. Imran, K. Saleem, A microservice recommendation mechanism based on mobile architecture, Journal of Network and Computer Applications 152 (2020) 102510.

[8] Z. He, P. Chen, X. Li, Y. Wang, G. Yu, C. Chen, X. Li, Z. Zheng, A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems, IEEE Transactions on Neural Networks and Learning Systems (2020) early access.

[9] M. Labiadh, C. Obrecht, C. F. da Silva, P. Ghodous, A microservice-based framework for exploring data selection in cross-building knowledge transfer, Service Oriented Computing and Applications (2020) 1–11.

[10] M. Jin, A. Lv, Y. Zhu, Z. Wen, Y. Zhong, Z. Zhao, J. Wu, H. Li, H. He, F. Chen, An anomaly detection algorithm for microservice architecture based on robust principal component analysis, IEEE Access 8 (2020) 226397–226480.

[11] W. Du, S. Ding, A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications, Artificial Intelligence Review (2020) 1–24.

[12] L. Yu, Y. Sun, Z. Xu, C. Shen, D. Yue, T. Jiang, X. Guan, Multi-agent deep reinforcement learning for hvac control in commercial buildings, IEEE Transactions on Smart Grid 12 (1) (2020) 407–419.

[13] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, R. Ke, Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous

driving, Transportation Research Part C: Emerging Technologies 117 (2020) 102662.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks 20 (1) (2008) 61–80.

[15] J. Jiao, W. Zhen, W. Zhu, G. Wang, Quality-related root cause diagnosis based on orthogonal kernel principal component regression and transfer entropy, IEEE Transactions on Industrial Informatics 17 (9) (2020) 6347–6356.

[16] Y. Wang, C. Zhao, S. Yang, X. Ren, L. Wang, P. Zhao, X. Yang, Mpcsm: Microservice placement for edge-cloud collaborative smart manufacturing, IEEE Transactions on Industrial Informatics 17 (9) 5898–5908.

[17] T. Zschörnig, J. Windolph, R. Wehlitz, B. Franczyk, A cloud-based analytics-platform for user-centric internet of things domains–prototype and performance evaluation, in: Proceedings of the 53rd Hawaii International Conference on System Sciences, 2020, pp. 6599–6608.

[18] D. Djenouri, R. Laidi, Y. Djenouri, I. Balasingham, Machine learning for smart building applications: Review and taxonomy, ACM Computing Surveys 52 (2) (2019) 1–36.

[19] R. Berta, A. Kobeissi, F. Bellotti, A. De Gloria, Atmosphere, an open source measurement-oriented data framework for iot, IEEE Transactions on Industrial Informatics 17 (3) (2020) 1927–1936.

[20] L. Meng, Y. Sun, S. Zhang, Midiag: A sequential trace-based fault diagnosis framework for microservices, in: International Conference on Services Computing, Springer, 2020, pp. 137–144.

[21] H. Chen, P. Chen, G. Yu, A framework of virtual war room and matrix sketch-based streaming anomaly detection for microservice systems, IEEE Access 8 (2020) 43413–43426.

[22] J. f. Cui, H. Xia, R. Zhang, B. x. Hu, X. g. Cheng, Optimization scheme for intrusion detection scheme gbdt in edge computing center, Computer Communications 168 (2021) 136–145.

[23] R. W. Collier, E. O'Neill, D. Lillis, G. O'Hare, Mams: Multi-agent microservices, in: The World Wide Web Conference, 2019, pp. 655–662.

[24] P. Krivic, P. Skocir, M. Kusek, G. Jezic, Microservices as agents in iot systems, in: KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications, Springer, 2017, pp. 22–31.

[25] A. A. Khaleq, I. Ra, Intelligent autoscaling of microservices in the cloud for real-time applications, IEEE Access 9 (2021) 35464–35476.

[26] S. T. Arzo, R. Bassoli, F. Granelli, F. H. Fitzek, Multi-agent based autonomic network management architecture, IEEE Transactions on Network and Service Management.

[27] Å. Hugo, B. Morin, K. Svantorp, Bridging mqtt and kafka to support c-its: a feasibility study, in: IEEE International Conference on Mobile Data Management, 2020, pp. 371–376.

[28] Z. Sun, C. Hu, C. Li, L. Wu, Domain ontology construction and evaluation for the entire process of software testing, IEEE Access 8 (2020) 205374–205385.

[29] Y. Djenouri, A. Belhadi, J. C.-W. Lin, A. Cano, Adapted k-nearest neighbors for detecting anomalies on spatio–temporal traffic flow, IEEE Access 7 (2019) 10015–10027.

[30] J. Zhang, M. Zulkernine, A. Haque, Random-forests-based network intrusion detection systems, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38 (5) (2008) 649–659.

[31] A. Zafari, R. Zurita-Milla, E. Izquierdo-Verdiguier, A multiscale random forest kernel for land cover classification, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13 (2020) 2842–2852.

[32] P. Karczmarek, A. Kiersztyn, W. Pedrycz, E. Al, K-means-based isolation forest, Knowledge-Based Systems 195 (2020) 105659.

[33] J. C. W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, M. Aloqaily, Privacy preserving multi-objective sanitization model in 6g iot environments, IEEE Internet of Things Journal (2020) early access.

[34] Y. Djenouri, G. Srivastava, J. C. W. Lin, Fast and accurate convolution neural network for detecting manufacturing data, IEEE Transactions on Industrial Informatics 17 (4) (2020) 2947–2955.