Microservices Adaptation using Machine Learning: A Systematic Mapping Study

Anouar Hilali, Hatim Hafiddi and Zineb El Akkaoui
InnovatiVE REsearch on Software, Systems and daTa,
Institut National des Postes et Télécommunications, Rabat, Morocco

Keywords: Self-adaptation, Microservices, Machine Learning.

Abstract: The Microservice architecture is increasingly becoming the preferred architecture of modern applications. The logically distinct components that make up microservices make continuous delivery easier compared to monolithic architectures. This feature however makes it difficult for engineers to control the underlying services and properly adapt them at run-time. Designing our microservices as self-adaptive systems helps us tackle this issue. Each microservice can then dynamically monitor and adapt its behavior to change certain aspects of itself to achieve self-adaptive goals. The use of statistical and Machine Learning (ML) techniques helps in this area in a lot of ways (e.g., predicting resource usage, anomaly detection, etc.). This paper aims to provide a state of the art of the use of ML in microservice adaptation, the main goal is to provide an overview of the field and identify the most frequent adaptation goals and the types of adaptation techniques used. In order to carry out a comprehensive analysis, a well-defined method of systematic mapping is performed to categorize, according to a detailed scheme, every paper relevant to this topic. The results can potentially shed light on areas where further investigation might be warranted.

1 INTRODUCTION

Nowadays, modern applications are changing from a monolithic architecture where the application is constructed as a single entity, to a more distributed architecture where complex applications are broken down into logically recognizable components: microservices (Khazaei et al., 2018a). The Microservice architecture is an architectural style in which an application is designed as a set of small services communicating with each other using a lightweight mechanism (Fowler, 2014). The decoupling of components that is characteristic of this type of architecture makes continuous delivery safer and cheaper contrary to many other architectural styles (Junior, 2018).

However, several obstacles appear that make it difficult to manually manage applications at run-time (e.g., resource optimization, granularity of services, context, fault detection, etc.). A solution to this is to design these microservices as self-adaptive systems that can dynamically monitor and adapt their behavior to change certain aspects of themselves to meet certain goals (e.g., when operating conditions are not stable and optimal (Mendonça et al., 2019), when services need to be discovered in a changing context (Wanigasekara, 2015a), etc.). Self-adaptation techniques are known to be promising ways of tackling and managing run-time uncertainties (Sanctis et al., 2020a). We can distinguish four adaptation goals in self-adaptive systems: self-configuration (i.e., systems that configure themselves automatically), self-optimization (systems that are constantly looking for ways to optimize performance), self-healing (systems that detect and fix anomalies) and self-protection (systems that defend themselves from attacks) (Khazaei et al., 2018a). An autonomic management system contains the logic that tackles one or several of these adaptation goals. The four elements: Monitor, Analyze, Plan, and Execute achieve the necessary functions of any self-adaptive system. These elements share common Knowledge therefore the model is usually referred to as the MAPE-K model (Kephart and Chess, 2003). Applying statistical and ML techniques in order to better any aspect of self-adaptive systems helps in bringing about adaptation without human intervention. ML algorithms for instance, can be employed to predict resource usage patterns based on historical data of an application’s microservices and effectively anticipate changes (e.g., auto-scaling, placement reconfiguration, etc.) (Junior, 2018).

The goal of this paper is to achieve a research area overview, identifying the most common microservices adaptation goals, techniques that are used in accomplishing these goals and the frequency of using ML algorithms in carrying this out. We therefore choose to perform a systematic mapping of the...
-existent literature. Generally, a systematic mapping study structures the type of research, reports and results that have been published by classifying them and generates a visual summary of its results (Kitchenman and Charters, 2007). In this work, the discussion of the findings focus on supporting the understanding of what has been addressed by the research community and eventually propose future research guidelines in this field.

The remainder of this paper is structured as follows: section 2 details the research methodology used for the mapping study accomplished in this work, from establishing the research questions to the resulting systematic map. Then, in section 3 we provide an analysis and a discussion of the results extracted during the mapping study. In section 4 we discuss the threats to validity of this paper. Section 5 specifies similar works and situates our contribution. Finally, section 6 outlines the main contribution of this paper and considers some directions that could be potentially investigated by the research community.

2 RESEARCH METHODOLOGY

In order to achieve the goals set above, our systematic mapping study process followed Kitchenman’s approach. This type of study is helpful when it is discovered that very little evidence is likely to exist on a certain topic (Kitchenman and Charters, 2007). Figure 1 represents the underlying steps of performing a systematic mapping study. Each step of the process has an outcome, the last step produces the systematic map. In the upcoming section we will describe each step in detail and apply it to our particular case.

2.1 Definition of Research Questions

Initially we must define the research scope by identifying goals that are expressed in research questions. We chose to follow structured criteria to frame our questions (Kitchenman and Charters, 2007), the following questions were stated:

**RQ1:** What are the microservices adaptation targets (i.e., sub-goals of those mentioned in the introduction) that the research community tackles?

**RQ2:** Are the adaptations on an application or an architecture level?

**RQ3:** What adaptation techniques were used?

Based on a classification scheme that we detail later on, we can answer more precise questions:

**RQ4:** If machine learning was used, what algorithms were picked?

**RQ5:** What level of empirical evidence was attained by researchers?

2.2 Conduct Search for Primary Studies

In order to answer our questions, we must identify the key studies by setting up a set of search strings (Petersen et al., 2008). Multiple digital libraries were chosen to extract relevant research using the two criteria suggested by Kitchenman: Population (i.e. An application area) and Intervention (a software methodology, tool, technology or procedure that addresses a specific issue) (Kitchenman and Charters, 2007):

- **Population:** Microservices.
- **Intervention:** Adaptation, self-adaptation, self-healing and context-aware.

Our Keywords can be deduced from each facet, the population and intervention facets lead us to a search string similar to this: (“microservices” OR “microservice” OR “micro-service” OR “micro-services”) AND (“adaptation” OR “adaptive” OR “self-adaptation” OR “self adaptation” OR “self-adaptive” OR “self adaptive” OR “context-aware” OR “context aware” OR “self-healing” OR “self healing”).

The search query slightly varies from one digital library to the other in order to comply with each library’s query rules.

The choice of what digital libraries to use was based on (Breton et al., 2007)’s identification of the most relevant digital libraries to software engineers. The choice of the search string was made in order
Table 1: Search queries used on different digital libraries and their respective results.

<table>
<thead>
<tr>
<th>Digital Library</th>
<th>Number of results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Digital Libray</td>
<td>16</td>
</tr>
<tr>
<td>IEEExplore</td>
<td>88</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>10</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>149</td>
</tr>
<tr>
<td>Google Scholar</td>
<td>12</td>
</tr>
</tbody>
</table>

to select all papers that discuss the adaptation of microservices regardless of tools used (e.g. Artificial Intelligence algorithms). We didn’t consider the comparison facet because it didn’t make sense in our case, while the outcome facet would restrict our research and wouldn’t allow us to paint a broad overview of the research area. The results are accurate as of 10th of January 2021, the number of total publications is 275. Table 1 shows the full results.

2.3 Screening of Papers for Inclusion and Exclusion

Inclusion and exclusion criteria were used to only focus on papers that are relevant to answering our domain questions. A percentage of these criteria are based on the fact that our main aim is microservice architectures. Other criteria are based on practical issues. The inclusion criteria were:

IC1: publications produced in English.
IC2: publications that are peer-reviewed and published in journals, conferences and workshops.
IC3: publications that explicitly mention our keywords in the title, abstract and/or keywords.

These exclusion criteria were used:

EC1: publications where the authors focus on architectures other than microservices.
EC2: publications such as books, chapters of books or similar ones.
EC3: publications that are not in the computer science field.

Duplicates are automatically dealt with using the merge duplicates feature on the open source research tool Zotero. Initially criteria IC1 was applied for all queries. No filters were used for IEEExplore. Criteria IC2, IC3, EC2 and EC3 were applied using the built-in filters of the mentioned digital libraries. "Research Article" and "Computer Science" filters were applied for ScienceDirect and ACM Digital Library and finally "Conference Paper" and "Article" filters were applied for SpringerLink. We then filtered through the publications manually, reading abstracts and if necessary introductions and conclusions. We ranked the papers based on their relevance to our objectives by labeling each paper with a number (0 would be irrelevant, 1 would be somewhat relevant and 2 as in relevant). We double checked the somewhat relevant papers a second time and discarded the irrelevant ones. Initially the number of relevant publications went from 275 to 129 based on reading abstracts. Then the final result was reduced to 62 publications that are useful for our purposes. Figure 2 illustrates this process.

2.4 Keywording of Relevant Papers

Before going ahead and setting up a classification scheme, the authors went through all the relevant papers and made sure they were all aligned with our objectives. We not only relied on reading the abstracts but at times had to carefully read the full paper. In order to properly cover all aspects of our research, the papers were classified using the following classification scheme:

- **Adaptation Targets (RQ1).** In order to answer RQ1, we classified papers based on what area of microservices were adapted.
- **Adaptation Techniques (RQ2).** To classify the techniques used in each research and address RQ2.
- **Adaptation Level (RQ3).** To tackle RQ3, we also classified papers based on what level of adaptation were the techniques aimed at.
- **Use of Machine Learning (RQ4).** To classify papers on whether they used machine learning in their adaptations and if yes, what type of algorithms were used in their solution.
- **Evidence (RQ5).** We chose an existing classification of research approaches to classify our papers (Wieringa et al., 2006) and address RQ5 on what type of empirical evidence is provided by each paper:
  - **Validation Research:** A paper where the techniques that are examined are innovative and are not yet applied in practice.
  - **Evaluation Research:** A paper where the techniques discussed were implemented in practice and an evaluation of these techniques are conducted.
  - **Solution Proposal:** A paper where a solution to a given problem is presented. It can be either new or built on top of existing approaches. The solution is also illustrated by a specific example.
  - **Philosophical Papers:** Papers where a novel perspective is proposed by structuring a given field.
  - **Opinion Papers:** Papers where the author express a personal opinion on the validity of a certain technique or how it should be implemented.
• *Experience Papers*: Papers where an explanation of how a technique has been done in practice is given. It relies on the personal experience of the authors.

### 2.5 Data Extraction and Mapping Process

We conclude our review with the data extraction step. All 64 papers relevant to us were fully read. Based on the research questions defined, an Excel data extraction table was developed to document the data extraction process. Thanks to the produced graphs, we are now able to have a proper overview of what the research community focuses on the microservices adaptation research area. Table 3 illustrates the detailed scheme.

### 3 MAIN FINDINGS

All articles were published between September 2015 and January 2021. Figure 3 shows an increase in the number of published articles on the topic of Microservice adaptation. The decrease between 2020 and 2021 is due to the fact that the results were as of the first month of 2021.

#### 3.1 RQ1: Adaptation Targets

Of all the texts that were reviewed, 42% (24 papers) tackled Resource Scheduling or Utilization and Placement, followed by discussions involving Auto-scaling and Elasticity at 21.0% (17 papers), then Context at 11.1% (9 papers), Fault/Anomaly Detection and Exception Handling at 9.9% (8 papers) and Granularity of Services at 3.7% (3 papers). Finally The Other section with 12.3% (11 papers) constitutes solutions tackling Orchestration/Collaboration, Task Workflow/Scheduling, Data Transfer/Filtering/Acquisition, Security/Fire-walling, Configuration/Recovery, Latency and Network Resource Consumption. Figure 4 illustrates the distribution of goals. Table 2 details each goal, its definition and the number of relevant papers.

We notice from the extracted data that there is a huge focus on optimizing for performance. A combined 63.0% between resource optimization and elasticity. We believe this might be due to the fact that industry professionals are mainly interested in optimizing resources and researchers follow through by proposing solutions for their concrete optimization issues.

#### 3.2 RQ2: Adaptation Level

Looking at the distribution of the adaptation level of papers, we notice that the majority of papers (37 at 59.7%) present a solution acting exclusively on an application level (i.e., solutions dealing with services, containers and applications). Followed by papers (16
Table 2: The different definitions of adaptation goals and the number of papers discussing them.

<table>
<thead>
<tr>
<th>Adaptation Target</th>
<th>Definition</th>
<th>Number of Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Scheduling/Utilization, Placement</td>
<td>Optimizing the use of resources (e.g., CPU, memory, bandwidth, etc.) and the adaptive placement of containers.</td>
<td>34</td>
</tr>
<tr>
<td>Auto-scaling and Elasticity</td>
<td>Automatically adjusting the capacity to insure steady performance.</td>
<td>17</td>
</tr>
<tr>
<td>Context</td>
<td>The ability to collect data about a system’s surrounding environment (e.g., Location, Temperature, etc.).</td>
<td>9</td>
</tr>
<tr>
<td>Fault/Anomaly Detection, Exception Handling</td>
<td>Detecting anomalies and errors in the faulty microservices and handling exceptions.</td>
<td>8</td>
</tr>
<tr>
<td>Granularity of Services</td>
<td>Identifying the optimal microservice boundaries.</td>
<td>3</td>
</tr>
<tr>
<td>Orchestration/Collaboration</td>
<td>Automatic coordination of containers.</td>
<td>2</td>
</tr>
<tr>
<td>Task Workflow/Scheduling</td>
<td>Assigning tasks to the proper resources.</td>
<td>2</td>
</tr>
<tr>
<td>Data Transfer/Filtering/Acquisition</td>
<td>The ability to manipulate data.</td>
<td>2</td>
</tr>
<tr>
<td>Security/Fire-walling</td>
<td>Protection against malicious attacks.</td>
<td>2</td>
</tr>
<tr>
<td>Configuration/Recovery</td>
<td>Having the proper system configuration (e.g., reliability, availability, performance, etc.) and the ability to recover after failing.</td>
<td>2</td>
</tr>
<tr>
<td>Latency</td>
<td>The delay for data to go from its source to its destination.</td>
<td>1</td>
</tr>
<tr>
<td>Network Resource Consumption</td>
<td>The underlying network’s resource consumption.</td>
<td>1</td>
</tr>
</tbody>
</table>

at 25.8%) with both an application and architectural (i.e., solutions dealing with how to design and architect your system) approach then purely architectural solutions with 9 papers (14.5%). Figure 5 illustrates this.

![Figure 5: The distribution of the adaptation level in relevant papers.](image)

Since 53 papers (85.5%) have an application level approach, we conclude that most research focuses on solutions that tackle issues on the level of services, applications or containers. This makes sense since there is a wide variety of possible contributions to have contrary to presenting solutions from an architectural and design optic.

### 3.3 RQ3: Adaptation Techniques

From the present study, we believe that custom techniques are the preferred approach for most researchers. In addition, ML algorithms (i.e., classical machine learning and deep learning) are the techniques of choice for numerous researchers in achieving adaptation targets. 10 papers employ a classical machine learning algorithm in their solution. Reinforcement learning, Deep learning and heuristics are used in 5 papers each. In 4 papers, researchers extend the existing Kubernetes tools. A modified autoscaler is introduced in 2 papers and so is the Non-Dominated Sorting Genetic Algorithm (in order to make a container allocation strategy and automatically manage elasticity). The remaining papers (19) have the following techniques:

- **Description language**: A custom language and an appropriate platform to aid in the implementation of self-adaptive microservices,
- **Resource scheduling algorithms**: Policies used to assign resources accordingly for energy efficiency purposes,
• **External services**: In-house developed software and delegating run-time management to the cloud,
• **Extended Berkeley Packet Filter**: Used to intercept key Linux system calls for efficient monitoring,
• **Scaling policy derivation tool**: Used to design and evaluate 5 auto-scaling policies,
• **Cloud-Edge-Dew architecture**: A custom architecture that optimizes the advantages of Cloud and Edge Computing to contribute microservices for end user devices,
• **Custom orchestrator**: A custom orchestrating tool for microservices,
• **Horizontal offloading mechanism**: A custom mechanism based on the OpenFog reference architecture where fog nodes can horizontally offload computations to multiple less loaded nodes,
• **Hierarchical scalable adaptive cloud monitoring architecture**: Custom architecture that monitors physical and virtual infrastructure, realizes scalability of monitoring based on microservices and adjusts the monitoring interval and data transmission strategy (Wang et al., 2020),
• **Latency aware microservice mashup algorithm**: A custom service mashup approach that focuses on network resource consumption in edge network in addition to efficient service mashup,
• **Micro-controllers**: A controller that can be composed or configured at run-time based on the adaptation goals of the target system,
• **Model based approach**: A custom method called Microservices Recovery Action Selection that adapts to frequent microservices changes without the need for historical data of previous failures,
• **Microservice ambients using aspects**: A modeling concept that considers microservice boundaries as an adaptable first-class entity, it is also based on the aspect-oriented architectural metamodeling approach of ambients (Hassan et al., 2017),
• **Architecture and planning engine at run-time**: An interactive and iterative planning engine that provides insight into which granularity adaptation strategy is adequate at run-time (Hassan, 2019),
• **Custom framework**: A self-adaptive root cause diagnosis framework in order to analyze several metrics collected from given microservices,
• **Event processing techniques**: Used as an integrated part to an appropriate architecture to better smart decision-making,

• **Application partitioning and task offloading algorithm**: Which comes up with an application partitioning decision at run-time,
• **Application-infrastructure co-programming model and architecture**: Which provides a controllable environment for the creation of application logic and enable reconfigurability of computing resources at run-time according to the needs of a specific application (Štefanić et al., 2019) and
• **Vehicular-OBUs-As-On-Demand-Fogs**: A custom framework that efficiently uses clusters of vehicles to benefit from on-board units in order to optimize resources.

### 3.4 RQ4: Use of Machine Learning

Here we notice that although small, there is a decent interest in using Artificial Intelligence in order to better microservice adaptation. 40.3% (25 papers) of papers utilize either classical machine learning or deep learning algorithms, the remaining 37 papers (59.7%) use other techniques mentioned in section 3.3. The following algorithms were used:

- Data Classification algorithms,
- Support Vector Machine,
- Bayesian Network,
- Decision Tree,
- Naive Bayes classifier,
- Contextual Bandit Reinforcement Learning,
- Stacked Long Short-Term Memory Networks,
- K-means,
- Planning algorithms,
- Extreme Learning Machine,
- Deep Q-Learning,
- Fuzzy Lattice Reasoning,
- Artificial Neural Network and
- Gaussian Process Regression.

Figure 6 illustrates the distribution of what targets researchers go to when utilizing ML as an adaptation technique. We notice that resource optimization is predominately the most common one with 53.1% of total papers that use ML.

### 3.5 RQ5: Evidence

The level of empirical evidence in the relevant papers is dominated by solution proposals at 54 papers...
Figure 6: Distribution of the most commonly tackled targets when using ML.

Figure 7: The distribution of types of papers.

(87.1%), followed by 4 (6.5%) opinion papers, 2 evaluation papers and 2 validation research papers (3.2%). Figure 7 shows the distribution.

This means that the research community is mainly focused on solving concrete microservice adaptation problems and supporting their solutions with real examples. The small number of papers (6 at 9.7%) where practical applications are not provided (i.e., validation research, philosophical, opinion and experience papers) indicates that there is virtually no to little to no interest in discussions that do not present a valid practical example.

4 THREATS TO VALIDITY

Assessing Threats to Validity is clinical in order to ensure quality empirical studies in Software Engineering (Zhou et al., 2016).

Construct Validity: The discussion surrounding our main findings in section 3 is only valid for the papers provided. We therefore made sure to include all the possible relevant papers. In order to realize this, we used all the databases that are relevant to software engineering research mentioned in section 2.2. We made sure to answer the appropriate research questions and deduce the adequate complete search terms, we for instance considered all the possible variations of our facets (e.g., microservices, self-adaptation). We also avoided the threat of inappropriate inclusion and exclusion criteria in our screening by considering the title, abstract and keywords. In addition, we decided as an initial safety measure not to exclude papers that haven’t been fully investigated until there has been an exhaustive reading.

Internal Validity: In order to alleviate the issues surrounding data extraction and classification, we took the proper time-frame to, at times, read introductions and conclusion and some times the entirety of a given paper if it wasn’t possible to extract data initially from the abstract. We also used an Excel table to properly store the extracted data to generate relevant statistics.

External Validity: We made sure not to restrict the time span of the resulting studies. To improve external validity in the future it is perhaps advised to have full access to all relevant papers by contacting their respective authors.

Conclusion Validity: The study’s replicability is possible thanks to the search method details provided in section 2. The threat of primary study duplication was also avoided by using the open search tool Zotero and always double checking our data.

5 RELATED WORK

Numerous systematic reviews tackled self-adaptation. One paper outlines the machine learning techniques used to approach self-adaptation based on the concerns, aspects and purposes of choice (Saputri and Lee, 2020), although exhaustive we believe it wasn’t applied to a specific type of architecture. Multiple papers discuss self-adaptation through the lens of a specific context (e.g., mobile app (Grua et al., 2019), cyber-physical system (Muccini et al., 2016), etc.). Although there is a discussion of self-adaptation in Service Oriented Architecture (Romay et al., 2011) there are no equivalent for the Microservices Architecture. To the best of our knowledge there were no Systematic Mapping Reviews discussing the use of ML techniques for the adaptation of microservices.

6 CONCLUSIONS

In this paper, we provide the results concerning the systematic mapping study of the use of ML in Microservice adaptation. this paper outlines important observations:

• recognizing the most focused on microservices
adaptation targets in the research community;
• highlighting the techniques of choice that authors go to;
• figuring out the level of empirical evidence achieved by the researchers’ presented solutions;

Our study, with the help of the developed classification scheme, provides the most commonly tackled issues in microservice adaptation targets; resource optimization and elasticity. It also indicates that although many authors decide to develop a unique adaptation technique, there is also a growing interest in using ML algorithms as techniques to achieve adaptation targets. Our findings also reflect that the majority of researchers provide practical solutions to concrete issues.

Our objective was to provide a comprehensive mapping of the use of ML in microservices adaptation. We intend to go more in depth and specify the strengths and weaknesses of specific techniques in specific contexts, explore opportunities where adaptation techniques can intersect and be more efficient at general self-adaptation and finally add to the growing discussion by introducing concepts that might help us make abstractions to adaptation targets and build self-adaptive microservices that comprise of all the main goals (i.e., self-configuration, self-optimization, self-healing and self-protection), which will be our future work.

REFERENCES
Symposium on Cluster, Cloud and Internet Computing (CCGRID), pages 290–299.


Self-Organizing Systems (SASO), pages 40–49. ISSN: 1949-3681.
APPENDIX

As mentioned in section 2.5, table 3 represents the detailed scheme we used to conduct our SMR.
<table>
<thead>
<tr>
<th>Number</th>
<th>Paper</th>
<th>Year</th>
<th>Target</th>
<th>Technique</th>
<th>Evidence</th>
<th>Use of ML</th>
<th>ML Evidence</th>
<th>Evaluation level</th>
<th>Year Evidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Zhang et al., 2018)</td>
<td>2018</td>
<td>Fresh detection</td>
<td>Multi-stream non-negative matrix factorization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>(Chen and Mahanti, 2019)</td>
<td>2019</td>
<td>Computer configuration, load prediction and scheduling, data transfer and filtering</td>
<td>Multi-stream non-negative matrix factorization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>(Zhang et al., 2019)</td>
<td>2019</td>
<td>Service and instance numbers</td>
<td>Multi-stream non-negative matrix factorization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>(Wang et al., 2019)</td>
<td>2019</td>
<td>Data acquisition</td>
<td>Big Data processing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>(Yu et al., 2020)</td>
<td>2020</td>
<td>Response scheduling</td>
<td>Resource scheduling (remote MIS/M integration), solar energy output (for the availability of renewable energy)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>(Rashid and Pampapaganna, 2013)</td>
<td>2013</td>
<td>Resource utilization</td>
<td>Resource scheduling (remote MIS/M integration), Solar energy output (for the availability of renewable energy)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>(Wu et al., 2018)</td>
<td>2018</td>
<td>Context</td>
<td>Contextual Reinforcement Learning algorithms</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>(Zhang et al., 2018)</td>
<td>2018</td>
<td>Response scaling, scheduling</td>
<td>Multi-stream non-negative matrix factorization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>(Khan et al., 2018b)</td>
<td>2018</td>
<td>Security, configuration, load prediction and scheduling optimization</td>
<td>Adaptation to external services</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>(Kumar and Tahan, 2019)</td>
<td>2019</td>
<td>SHA-based session allocation</td>
<td>Multi-stream non-negative matrix factorization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>(Condon et al., 2020)</td>
<td>2020</td>
<td>Resource and task scheduling</td>
<td>Stacked LSTM Short-Term Memory Network</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>(Pall et al., 2017)</td>
<td>2017</td>
<td>Application- and user-level</td>
<td>Graph-based algorithms, k-means</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>(Ricci et al., 2020)</td>
<td>2020</td>
<td>Automatically detection</td>
<td>Machine learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>(Boll et al., 2017)</td>
<td>2017</td>
<td>Context and application</td>
<td>Service-oriented middleware</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>(Ghasemzadeh and Saberi, 2019)</td>
<td>2019</td>
<td>Resource and placement</td>
<td>Neural-based load prediction for placement and data-driven automatic deployment</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>(Shinha et al., 2020)</td>
<td>2020</td>
<td>Resource and auto-scaling</td>
<td>Reinforcement auto-scaler and Belief Rate</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>17</td>
<td>(Nur et al., 2020)</td>
<td>2020</td>
<td>Adoptive deployment</td>
<td>Extended Berkeley Packet Filter</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>18</td>
<td>(Ramirez et al., 2019)</td>
<td>2019</td>
<td>Auto-scaling</td>
<td>Scaling Policy Derivation Tool</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>19</td>
<td>(Khim et al., 2018)</td>
<td>2018</td>
<td>Resource and auto-scaling</td>
<td>Heuristics</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>(Wang et al., 2018)</td>
<td>2018</td>
<td>Exception handling</td>
<td>AI Planning algorithms</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>(Ghasemzadeh and Saberi, 2019)</td>
<td>2019</td>
<td>Machine Learning</td>
<td>Machine Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>22</td>
<td>(Takemoto et al., 2019)</td>
<td>2019</td>
<td>Latency</td>
<td>Multi-layer Peer-2-Peer Architecture</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>23</td>
<td>(Kumar and Singh, 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Machine Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>24</td>
<td>(Magalhães, 2016)</td>
<td>2016</td>
<td>Anomaly detection</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>25</td>
<td>(Kumar et al., 2017)</td>
<td>2017</td>
<td>Auto-scaling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>26</td>
<td>(Abdullah et al., 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Deep Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>27</td>
<td>(Xu et al., 2020)</td>
<td>2020</td>
<td>Resource and task scheduling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>28</td>
<td>(De Senchou et al., 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>(Kim et al., 2019)</td>
<td>2019</td>
<td>Context</td>
<td>Machine Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>(Khim et al., 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Machine Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>31</td>
<td>(Podolský et al., 2019)</td>
<td>2019</td>
<td>Auto-scaling, resource</td>
<td>Machine Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>(Hassanzadeh et al., 2019a)</td>
<td>2019</td>
<td>Resource and energy efficiency management</td>
<td>Non-dominated Sorting Genetic Algorithm</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>33</td>
<td>(Taheri and Malek, 2020)</td>
<td>2020</td>
<td>Auto-scaling, resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>34</td>
<td>(Zubia and Vazquez, 2017)</td>
<td>2017</td>
<td>Auto-scaling</td>
<td>Heuristics</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>35</td>
<td>(Rivco et al., 2019a)</td>
<td>2019</td>
<td>Elasticity, Auto-scaling</td>
<td>Cloud-based Service Delivery Platform</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>36</td>
<td>(Mittal et al., 2018)</td>
<td>2018</td>
<td>Context</td>
<td>Heuristics</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>37</td>
<td>(Wang et al., 2018)</td>
<td>2018</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>38</td>
<td>(Samanta et al., 2019)</td>
<td>2019</td>
<td>Resource, placement</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>39</td>
<td>(Elhajj et al., 2020)</td>
<td>2020</td>
<td>Network resource allocation</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>40</td>
<td>(Johndark et al., 2020)</td>
<td>2020</td>
<td>Auto-scaling, resource</td>
<td>LSTM on different data types</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>41</td>
<td>(Narayana et al., 2020)</td>
<td>2020</td>
<td>Context</td>
<td>Machine learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>42</td>
<td>(Wu et al., 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>43</td>
<td>(Hussain et al., 2017)</td>
<td>2017</td>
<td>Security</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>44</td>
<td>(Rahman et al., 2019)</td>
<td>2019</td>
<td>Context</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>45</td>
<td>(Hassanzadeh et al., 2019b)</td>
<td>2019</td>
<td>Context</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>46</td>
<td>(Yu et al., 2019)</td>
<td>2019</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>47</td>
<td>(Hansin et al., 2019)</td>
<td>2019</td>
<td>Context</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>48</td>
<td>(Nguyen and Nibbalin, 2017)</td>
<td>2017</td>
<td>Resource, scheduling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>49</td>
<td>(Ma et al., 2019)</td>
<td>2019</td>
<td>Anomaly detection</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>50</td>
<td>(Narayana et al., 2020)</td>
<td>2020</td>
<td>Context</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>51</td>
<td>(Oza et al., 2019)</td>
<td>2019</td>
<td>Context</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>52</td>
<td>(Ganapathi et al., 2018)</td>
<td>2018</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>53</td>
<td>(Kang and Latre, 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>54</td>
<td>(Rossi et al., 2020a)</td>
<td>2020</td>
<td>Auto-scaling, resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>55</td>
<td>(Yadav et al., 2020)</td>
<td>2020</td>
<td>Resource</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>56</td>
<td>(Chen et al., 2020)</td>
<td>2020</td>
<td>Resource, auto-scaling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>57</td>
<td>(Safarani et al., 2019)</td>
<td>2019</td>
<td>Resource, auto-scaling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>58</td>
<td>(Hee and Moh, 2020)</td>
<td>2020</td>
<td>Resource, auto-scaling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>59</td>
<td>(Wang, 2019)</td>
<td>2019</td>
<td>Resource, auto-scaling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>60</td>
<td>(Lakhan and Lau, 2020)</td>
<td>2020</td>
<td>Resource, auto-scaling</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>61</td>
<td>(Kostygova et al., 2019)</td>
<td>2019</td>
<td>Context</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>62</td>
<td>(Kara et al., 2020)</td>
<td>2020</td>
<td>Resource, Content</td>
<td>Reinforcement Learning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>