informatics

MDPI

Article

Reinforcement Learning for Reducing the Interruptions and Increasing Fault Tolerance in the Cloud Environment

Prathamesh Lahande, Parag Kaveri and Jatinderkumar Saini *

Symbiosis Institute of Computer Studies and Research, Symbiosis International (Deemed University), Pune 411016, India; prathamesh.lahande@sicsr.ac.in (P.L.); parag.kaveri@sicsr.ac.in (P.K.)
* Correspondence: saini_expert@yahoo.com

Abstract: Cloud computing delivers robust computational services by processing tasks on its virtual machines (VMs) using resource-scheduling algorithms. The cloud's existing algorithms provide limited results due to inappropriate resource scheduling. Additionally, these algorithms cannot process tasks generating faults while being computed. The primary reason for this is that these existing algorithms need an intelligence mechanism to enhance their abilities. To provide an intelligence mechanism to improve the resource-scheduling process and provision the fault-tolerance mechanism, an algorithm named reinforcement learning-shortest job first (RL-SJF) has been implemented by integrating the RL technique with the existing SJF algorithm. An experiment was conducted in a simulation platform to compare the working of RL-SJF with SJF, and challenging tasks were computed in multiple scenarios. The experimental results convey that the RL-SJF algorithm enhances the resource-scheduling process by improving the aggregate cost by 14.88% compared to the SJF algorithm. Additionally, the RL-SJF algorithm provided a fault-tolerance mechanism by computing 55.52% of the total tasks compared to 11.11% of the SJF algorithm. Thus, the RL-SJF algorithm improves the overall cloud performance and provides the ideal quality of service (QoS).

Keywords: cloud-computing; performance; reinforcement learning

Citation: Lahande, P.; Kaveri, P.; Saini, J. Reinforcement Learning for Reducing the Interruptions and Increasing Fault Tolerance in the Cloud Environment. *Informatics* **2023**, *10*, 64. https://doi.org/10.3390/informatics10030064

Academic Editors: Pavel Lyakhov and Maxim Deryabin

Received: 26 May 2023
Revised: 21 July 2023
Accepted: 31 July 2023
Published: 2 August 2023

1. Introduction

The cloud hosts a network of remote servers to provide a platform to compute various challenging user tasks [1]. To compute these tasks on its virtual machines (VMs), the cloud environment relies mainly on its resource-scheduling algorithms to provide the ideal expected results [2,3]. Since the existing scheduling algorithms are not provisioned with any external intelligence, they cannot dynamically adapt to the current scenario of the cloud at any given instance, leading to improper resource scheduling and limited results [2,3]. Additionally, the cloud experiences damage when the currently computed task generates uncertain faults, such as breaches of cloud security, violations of service level agreements (SLAs), and data loss [2–4]. This leads to the cloud being vulnerable and fault-intolerant. To focus on these issues, an intelligence mechanism is provided by integrating the reinforcement learning (RL) [5] technique with the existing resource-scheduling algorithm shortest job first (SJF) to design and implement an algorithm, RL-SJF. The RL-SJF algorithm enhances the resource-scheduling process and provides the much-needed fault-tolerance mechanism to the cloud. The primary reason for using the RL method integrated with the SJF is that its mechanism and work are close to how human beings learn.

The main reason for choosing the SJF algorithm is that this algorithm provides the ideal and best results among the cloud-scheduling algorithms with respect to the time parameters and also since it is very handy for long-term scheduling [6,7]. Hence, it has been combined with the RL method. The proposed RL-SJF algorithm has been implemented in a cloud-simulated environment where challenging tasks are computed in several scenarios and

circumstances. To fairly compare the results of RL-SJF with the existing SJF algorithm, tasks computed by the RL-SJF were computed using the bare SJF algorithm also. Since the existing SJF algorithm lacks decision-making, it cannot adapt to the cloud's current situations at any given instance. On the contrary, with ideal feedback mechanisms with every action, the RL-SJF algorithm enhances its decision-making by allocating the suitable VM to every task. Thus, the overall cost required to process all tasks is reduced with improved resource scheduling and better QoS will be provided to the end-user. Additionally, the RL-SJF algorithm handles the dynamically occurring faults while being computed and provides a solution to these faults. Thus, the performance of the cloud is enhanced over time with RL-SJF, thereby providing better cost and resource-scheduling results while keeping the cloud safe and secure from all faults. Figure 1 depicts the mechanism of computing the user tasks using the existing SJF and the proposed RL-SJF algorithms.

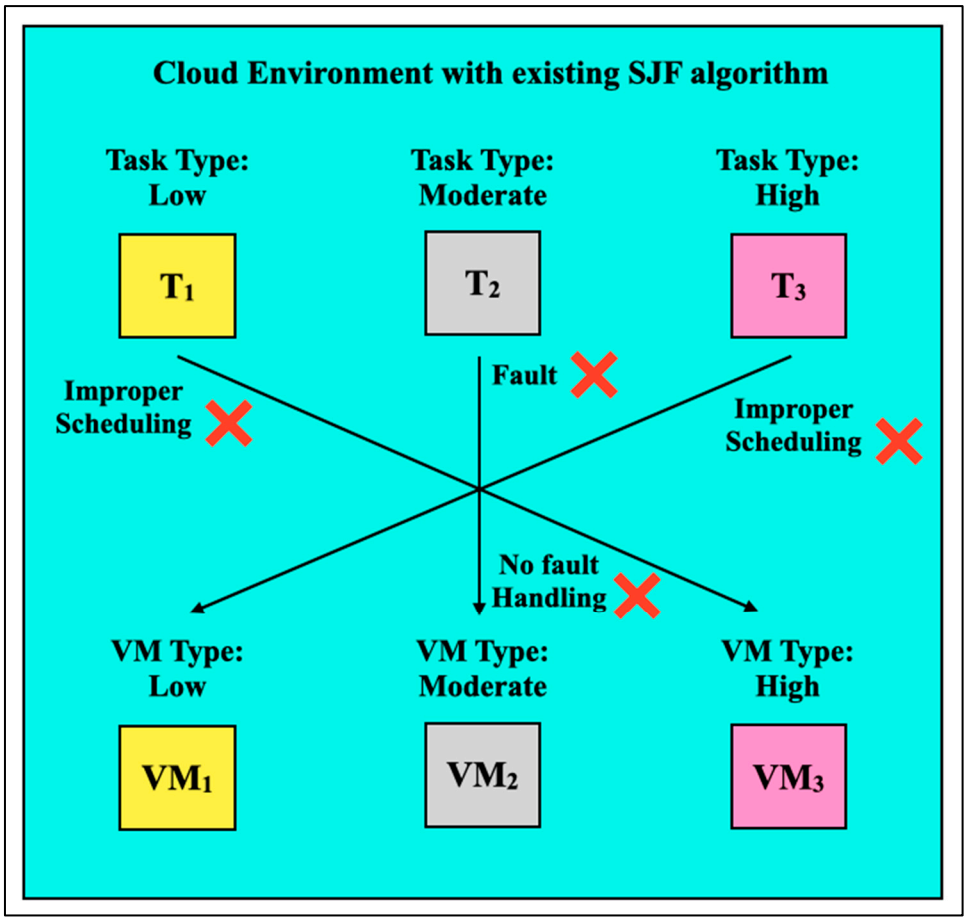

Figure 1. Mechanism of computing tasks using the existing SJF algorithm.

From Figure 1, we can observe that in the cloud environment with the existing SJF algorithm, there is a mismatch between the task to be computed and the VM it has been allotted to for computations. Additionally, some fault-generating tasks are not handled by the SJF algorithm. Since the mechanism of RL has been combined with this SJF algorithm to design and implement the RL-SJF algorithm, this algorithm undergoes several trial-and-error processes and obtains a series of corresponding rewards for its scheduling processes. This can be observed from Figure 2. With time and proper rewards, the RL-SJF can make ideal scheduling decisions in the cloud environment, ensuring its resources are utilized to their maximum. Additionally, the fault-generating tasks are handled by the RL-SJF algorithm by providing a solution to those faults and making sure the task has been computed without hampering the cloud's performance.

The rest of the paper is organized as follows:

- Related Works provided in Section 2.
- Experimental Design provided in Section 3.
- Results and Implications provided in Section 4.
- Validating Experimental Results using Empirical Analysis provided in Section 5.
- Conclusion provided in Section 6.

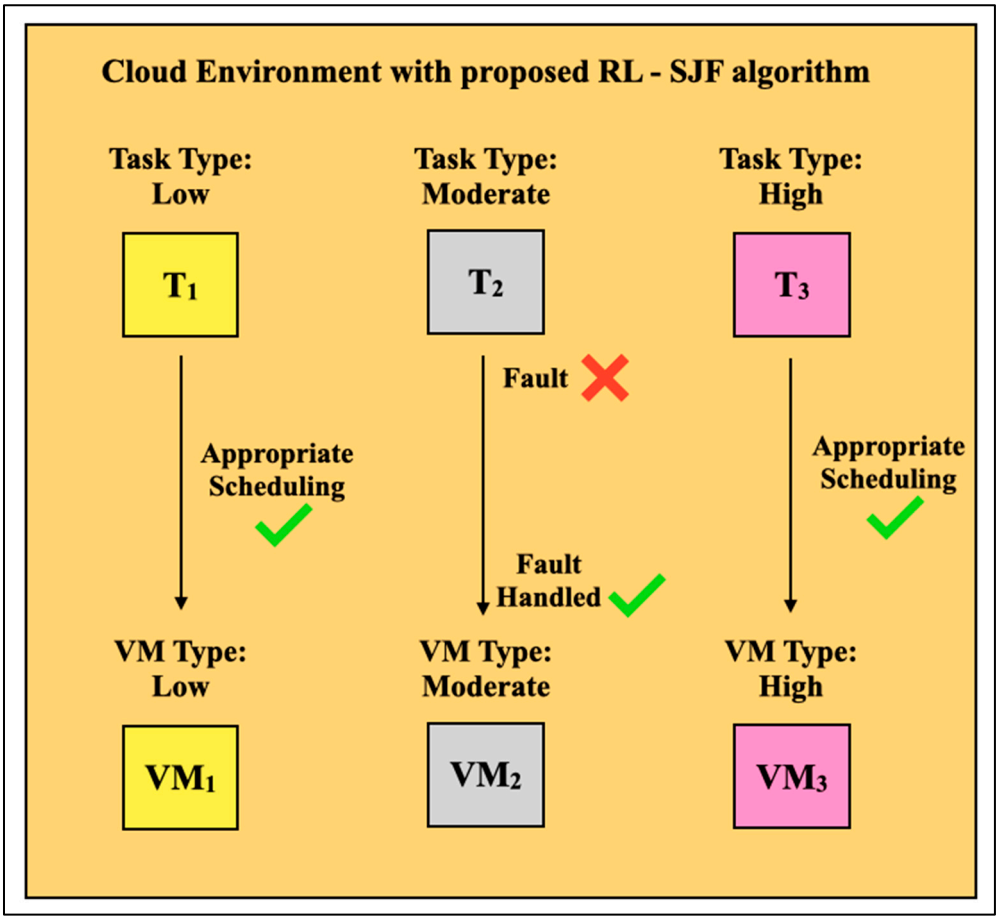

Figure 2. Mechanism of computing tasks using the proposed RL-SJF algorithm.

2. Related Works

The RL method has been used not only to focus on issues in the cloud computing environment but also to solve any challenge in other domains around the cloud. Several researchers have used this RL method to give an intelligence mechanism to the cloud by designing and implementing various strategies, algorithms, and techniques to improve cloud performance. The researchers in this paper have used the RL method intending to improve the resource-allocation strategies and overall network latencies in the cloud-edge environment [6]. The improvements made in allocating the resources mainly depend upon the past request history and states of the cloud networks. The simulation results convey that the presented method using RL has lower latencies compared with the existing solutions in the cloud-edge environment. To focus on the issue of job-shop scheduling (JSS), the researchers have used the RL method to improve the job-scheduling process [8]. The performance of this method is found to be better when compared with the existing benchmarked methods. To address the issues of energy consumption and latencies of the task, the researchers have presented a task offloading (TO) strategy based on RL, which is adaptive and takes little time to adjust to newer environments [9]. The experiment results convey optimal energy consumption while reducing task-processing delays and energy consumption.

To improve the overall performance in the cloud environment, this study has focused on detecting various anomalies in the surveillance of videos [10]. The experimental results convey multiple anomalies were detected with higher accuracy when compared with the existing studies. Regarding the issues of resource allocation, where a task is allotted to a VM, the researchers have proposed a task-scheduling approach using the RL method [11]. This approach outperforms the existing ones in the overall resource-allocation and task-scheduling approaches, improving the successful computing ratio and overall task computing satisfaction level. The researchers have used the RL method to solve the routing problems in a network environment [12]. This improvement in routing lowers the required time, cost, and bandwidth. The RL method has been implemented to solve the traveling salesman problem (TSP) and presents it as a dynamic TSP [13]. The DTSP adapts to the environment's dynamic changes and provides more satisfactory results of more than five percent more than the traditional TSP within less time. This paper used the RL method to solve the mathematical issue of the knapsack problem (KP) [14]. The process of selecting items is observed to be constructive, flexible, and adaptive with the RL technique concerning the cost required. This method provides better results concerning the existing greedy method.

A hybrid approach combining the RL and constraint programming has been presented in this research paper to provide a solution to the TSP, management of the portfolio, and KP [15]. The experimental results prove this strategy provides enhanced output compared to existing ones. The Q-learning (QL) technique of the RL has been implemented to improve the cache problem in the internet-of-things (IoT) environment [16]. The performance improvements made in this cache provide better results regarding response time and bandwidth requirements. The researchers have applied the RL algorithms QL, state-action-reward-state-action (SARSA), Watkin's Q(λ), and SARSA(λ) for single-machine scheduling in the online environment [17]. The experiment results convey that Watkin's Q(λ) provides enhanced results compared with the rest of the RL algorithms. To improve the JSS issue, this paper uses the RL method [18]. The experimental results convey better results by enhancing the JSS and providing a better balance between the response and computing times. This study presents an edge-caching technique based on RL to improve the ability to adjust dynamically in the IoT environment [19]. This method reduces the performance losses, makespan, traffic, and, most importantly, hit rates in the cache. The researchers in this paper have focused on the JSS problem and used the RL method and graphs' neural networks to solve it [20]. The experimental results provide better results when compared with the existing ones.

To focus on the response time of the resource-scheduling problem, the researchers have used a combined communication and computational resource-allocation strategy to optimize the task scheduling process in the cloud-computing environment [21]. The results convey better latencies when compared with the existing ones. The researchers have used an architecture consisting of multiple rewards, which is based on RL, to provide a solution to the highway driving problems [22]. The experimental results depict that this method provides better results when compared with existing ones concerning speed, frequency of exchanging lanes, and safety issues. The RL method has been implemented along with the Q-Learning method in the MEC environment to tackle resource allocation issues [23]. The results convey that this presented method achieves better cost when contrasted to the existing algorithms. The problems of resource scheduling in the network environments of radio access have been focused on in this research paper, and the RL method has been used to tackle these issues [24]. This method shows comparatively higher effectiveness, since the RL process accelerates the system's learning process.

The RL method, which learns from past experiences, has been used to tackle the issues in the wireless local area network [25]. The experimental results are verified and validated by comparing them with the current developments. To reduce the traffic in wireless networks, the researchers have focused on the cache-enhancement problem using the RL method [26]. This strategy improves the long-term and short-term cache hit rate

compared to the other existing methods. This paper focuses on using the RL technique for performing predictive analysis in smart cities in a deterministic environment [27]. The researchers have used the RL method to improve the overall reliability in managing resources in distributed systems [28]. The experimental results convey improvement in the resource allotment and effectiveness of using the RL method. Researchers have used the RL method to enhance the scheduling process in the JSS issue [29]. This paper enhances the resource-scheduling process and evaluates its efficiency using benchmark solutions. The RL method has been implemented to make the overall resource-scheduling dynamic [30]. This method provides enhanced resource allocation along with a reduction in migration costs. The network slicing problem, which aims to map numerous services to a single shared network infrastructure and distribute network resources to meet various quality of service (QoS) needs, has been taken into consideration by the authors of this study [31]. They have also presented a formulation approach to address this issue. The superiority of the proposed formulations over current ones is shown by numerical findings. An approach for predicting experience quality parameters in a larger communication system, including users and a communications network, based on the predicted values of QoS indicators, is described [32]. There are four normalization methods discussed, which are suggested to normalize an indicator's scale. The software-defined-networking idea is expanded in this paper to include wireless networks [33]. In this study, the mathematical model that enabled the authors to obtain this parameter is provided and reviewed. Finally, by putting the suggested solutions into practice, their usefulness has been assessed.

3. Experimental Design

This section includes the detailed design of the experiment conducted, which is further presented in three sub-sections: Section 3.1 provides the detailed experimental configurations related to the simulation environment. Section 3.2 includes the proposed RL-SJF algorithm. Section 3.1 also provides the dataset used for computations by the SJF and RL-SJF algorithms during the experiment.

3.1. Configuring the Simulation Environment

The simulation platform of the WorkflowSim [34] environment has been used for configuring the cloud-computing environment. The algorithms SJF and RL-SJF have been incorporated into this simulation platform. The SJF mechanism of choosing tasks from the cloud's ready queue having the least VM computational time is followed in the RL-SJF algorithm. However, the major change between SJF and RL-SJF is that the RL-SJF algorithm possesses intelligence to enhance the resource scheduling and provide fault tolerance to the cloud. This experiment has been conducted into two phases: **Phase I**: Computing all the tasks using the existing SJF algorithm; **Phase II**: Computing all the tasks using the proposed RL-SJF algorithm. To fairly compare the behavior and mechanism of the RL-SJF algorithm with the SJF algorithm under various scenarios, circumstances, and conditions, both these phases consist of fifteen scenarios, where the number of VMs that compute the tasks starts at five and increases by five VMs in each subsequent scenario until the tenth scenario, consisting of fifty VMs. Following this, the number of VMs for the eleventh scenario is a hundred, increasing by one hundred until the fifteenth scenario, in which the number of VMs is five hundred.

The scenarios considered for each phase can be represented as follows:

- **Scenario i**: Number of VMs = 5;
- **Scenario ii**: Number of VMs = 10;
- **Scenario iii**: Number of VMs = 15; and so on until,
- **Scenario x**: Number of VMs = 50; where number of VMs increase by count of 5.
- **Scenario xi**: Number of VMs = 100;
- **Scenario xii**: Number of VMs = 200;
- **Scenario xiii**: Number of VMs = 300;
- **Scenario xiv**: Number of VMs = 400;

- **Scenario xv**: Number of VMs = 500; where number of VMs increase by count of 100.

The task-event dataset provided by Alibaba has been used for computing tasks by both the SJF and RL-SJF algorithms. Here, each task consists of the following:

- **Task ID:** representing a unique number to identify a certain task;
- **Planned CPU**: represents the task's total computing time
- **Task Type**: Low (L) ..... if $10 \leq$ Planned CPU ≤ 60
  Medium (M) ..... if $70 \leq$ Planned CPU ≤ 300
  High (H) ..... if Planned CPU > 300

The dataset consisting of tasks to be computed differs from scenario to scenario. However, the dataset used for a certain scenario for SJF has also been used for the same scenario for RL-SJF. The primary reason for doing this is to fairly compare the RL-SJF algorithm with the SJF algorithm scenario-wise. Any task being computed on the cloud VM can either be computed without any faults or it may generate faults to hamper the cloud's performance. For such tasks, if the RL-SJF algorithm makes poor scheduling decisions regarding either scheduling or faults, it is provided with negative rewards; similarly, with ideal scheduling decisions and fault tolerance, it is provided with positive rewards. Here, with faults, the RL-SJF will be initially provided with a solution so that the system can use the same solutions for providing a fault-tolerance mechanism. Therefore, for any scheduling or fault-tolerating purposes, the RL-SJF algorithm monitors the previous reward obtained and accordingly improves its decision. With this, the RL-SJF algorithm provides the much-needed intelligence mechanism to provision a fault-tolerance mechanism and computes that task successfully. The various faults considered for this study are shown in Table 1.

Table 1. Task faults with its description.

Sr. No.	Task Fault	Description of the Task Fault
1	Unavailability of VMs	No free VMs available to compute tasks at a certain instance of time.
2	Breaching the cloud security	The currently computing task on a certain VM breaches the security of the cloud.
3	All VMs are deadlocked	A situation where all the VMs are blocked because each VM is holding a cloud resource and waiting for another resource held by another VM.
4	Task denied computing service	A certain task has been waiting to be computed in the cloud's task queue and has suffered from starvation.
5	Data loss observed at the cloud	The currently computing task on a certain VM accidently or intentionally causes a data loss at the cloud end.
6	Cloud accounts hijacked	The currently computing task on a certain VM intentionally hacks cloud accounts.
7	Cloud's SLAs violations	The currently computing task on a certain VM violates the regulatory measures mentioned in the SLAs.
8	Insufficient RAM	RAM of the VM is low on memory

The cloud VMs which process and compute the tasks are divided into nine categories according to their configurations. The categories are: L-L, L-M, L-H, M-L, M-M, M-H, H-L, H-M, and H-H. The VMs in every scenario are divided in such a way that a greater number of 'M' configuration VMs are there so that these VMs can be available to process and compute a greater number of tasks. The RL-SJF algorithm uses rewards as feedback to the cloud environment to enhance its decision-making process. The RL-SJF algorithm uses Q-Table to manage these rewards with every action it performs. In case of improper scheduling and/or faults thrown up by the tasks, a lower reward is generated by the RL-SJF algorithm, indicating that the decision-making needs improvements. A high reward is generated when the scheduling is appropriate, and the task is processed without faults. This process of reward offering continues over a period of time, and with time the decision-making ability of the RL-SJF algorithm enhances, thereby enhancing the scheduling process and provisioning the fault-tolerance mechanism.

3.2. Architecture of the RL-SJF and Working of Q-Table with RL-SJF

Figure 3 depicts the entire architecture of the proposed algorithm.

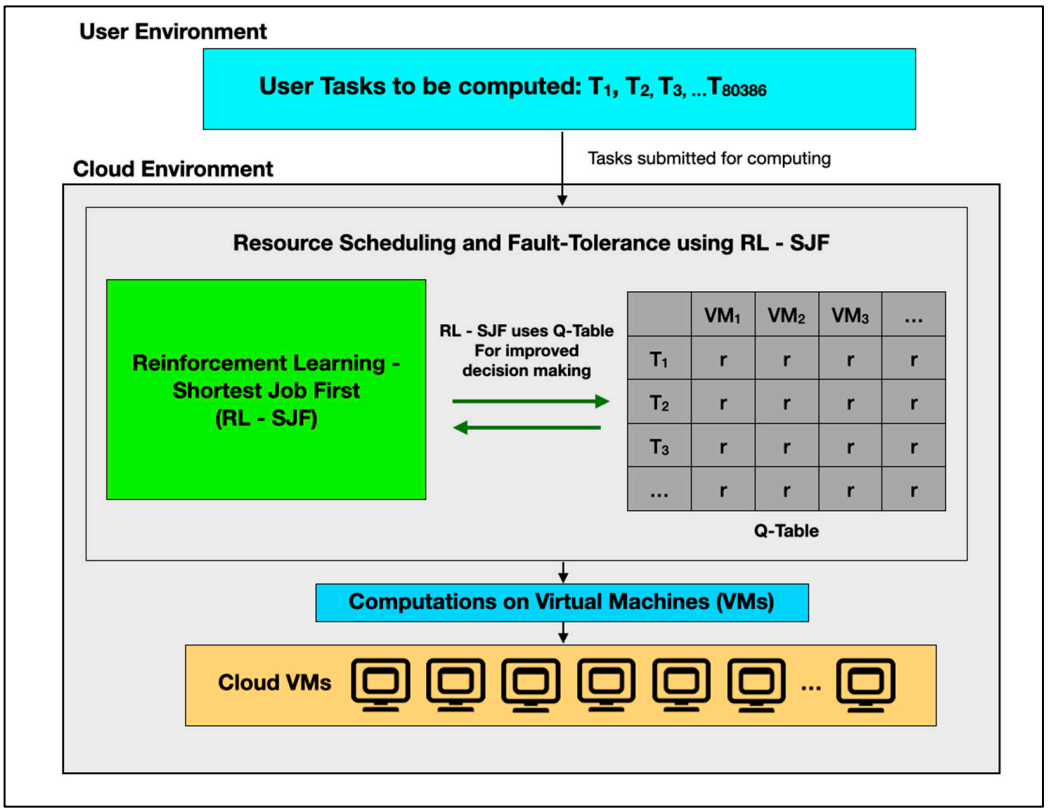

Figure 3. Architecture of the proposed RL-SJF algorithm.

The user tasks at the users' end are submitted to the cloud for computing. The cloud accepts them and puts them in a queue. Since the proposed algorithm is a blend of RL and SJF, the RL-SJF algorithm schedules tasks with minimum completion time, and additionally, RL provides the intelligence mechanism. While allocating the appropriate VM to a certain task, the RL-SJF takes inputs from the Q-Table in the form of rewards. The Q-Table is an additional data structure maintained in this architecture to store all the rewards where every task has an associated reward for a VM. The rewards are offered as follows:

- Task's Planned CPU 'Low' allotted to 'Low' performance VM: Highest Ideal Reward
- Task's Planned CPU 'Medium' allotted to 'Low' performance VM: Medium-Low Reward
- Task's Planned CPU 'High' allotted to 'Low' performance VM: Lowest Reward
- Task's Planned CPU 'Low' allotted to 'Medium' performance VM: Low-Medium Reward
- Task's Planned CPU 'Medium' allotted to 'Medium' performance VM: Highest Ideal Reward
- Task's Planned CPU 'High' allotted to 'Medium' performance VM: High-Medium Reward
- Task's Planned CPU 'Low' allotted to 'High' performance VM: Lowest Reward
- Task's Planned CPU 'Medium' allotted to 'High' performance VM: Medium-High Reward
- Task's Planned CPU 'High' allotted to 'High' performance VM: Highest Ideal Reward

With every task scheduling to any VM, the RL-SJF algorithm takes input from this Q-Table and keeps updating it dynamically. Here, the current task to be computed is taken from the queue considering its shortest, i.e., lowest computational time. When any next task has to be scheduled, the RL-SJF algorithm finds the currently vacant VM and calculates

the reward. Similarly, it obtains the highest reward for that task from the Q-Table and obtains the corresponding VM. Later, by comparing the currently vacant VM with the ideal VM, the reward is updated, and accordingly, scheduling and fault-tolerance mechanism is performed. Over a period of time, the Q-Table will possess enhanced rewards, thereby helping the RL-SJF to make better decisions and improving the overall cloud performance.

4. Results and Their Implications

This section includes the experimental results and their implications, presented into two sub-sections: Section 4.1 consists of the experimental results concerning the resource-scheduling process, and Section 4.2 contains the experimental results regarding the fault-tolerance mechanism.

4.1. Resource-Scheduling Results

This sub-section includes the experimental results of the algorithms SJF and RL-SJF concerning resource-scheduling mechanisms. The improvements in the resource-scheduling process can be observed by considering the aggregate required cost scenario-wise to compute the tasks by the algorithms. To have a considerate and fair comparison between the existing SJF and the proposed RL-SJF algorithm, the tasks that the SJF algorithm has successfully computed are computed using the RL-SJF algorithm. Table 2 represents the cost comparison of SJF and RL-SJF algorithms across all the scenarios.

Table 2. Cost comparison of SJF and RL-SJF across all the scenarios.

Scenario	VMs	SJF (in $)	RL-SJF (in $)	Decrease in Cost Percentage (in %)	Performance Comparison
1	5	17.05	16.76	1.71	RL-SJF > SJF
2	10	29.25	28.04	4.14	RL-SJF > SJF
3	15	38.66	35.11	9.19	RL-SJF > SJF
4	20	45.67	39.83	12.79	RL-SJF > SJF
5	25	51.02	43.09	15.55	RL-SJF > SJF
6	30	54.74	44.82	18.13	RL-SJF > SJF
7	35	57.28	45.95	19.79	RL-SJF > SJF
8	40	59.68	47.05	21.17	RL-SJF > SJF
9	45	61.35	47.48	22.61	RL-SJF > SJF
10	50	62.68	47.85	23.66	RL-SJF > SJF
11	100	73.71	56.12	23.87	RL-SJF > SJF
12	200	78.43	59.12	24.62	RL-SJF > SJF
13	300	83.14	62.12	25.28	RL-SJF > SJF
14	400	87.87	65.13	25.88	RL-SJF > SJF
15	500	92.59	68.13	26.41	RL-SJF > SJF
Average		59.54	47.11	18.32	18.32

From Table 2, the following observations can be made across all scenarios:

- ↑ VM = ↑ Cost: The overall cost required rises with the number of VMs.
- Total Cost (SJF) = $893.12
- Total Cost (RL-SJF) = $706.6.
- Average Cost (SJF) = $59.54.
- Average Cost (RL-SJF) = $47.11.
- Average Decrease in Cost Percentage = 18.32%
- Performance (RL-SJF) > Performance (SJF) concerning resource scheduling across all the scenarios.
- Reduction in the cost percentage in the above table signifies how much cost is saved by the RL-SJF algorithm as compared to that of SJF.

4.2. Fault-Tolerance Results

This Section 4.2 presents the experimental results and their implications concerning the fault-tolerant mechanism. The existing SJF algorithm cannot compute or process any tasks causing faults since it does not have any external intelligence provided to it. On the other hand, the proposed RL-SJF algorithm intelligently provides a solution to the tasks that generated faults, such as the unavailability of VMs, VMs in the deadlocked state, services denied to a task, and insufficiency of RAM, and computes them successfully. Additionally, faults such as security breaches, loss of data, hijacked accounts, and violations of SLAs are intelligently tracked and not computed by the RL-SJF to ensure the cloud is not damaged and is secure.

Table 3 represents the status of the tasks which have been computed or not with respect to the existing SJF.

Table 3. Status of tasks computed by SJF algorithm across all scenarios.

	SJF														
	Number of Virtual Machines (VMs)														
Task Fault	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**100**	**200**	**300**	**400**	**500**
VMs unavailable	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Security Breach	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Deadlocked VMs	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Service Denied	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Loss of Data	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Hijacked Accounts	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Violations of SLAs	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Insufficient RAM	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
No Fault	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

Table 4 represents the status of the tasks which have been computed or not with respect to the proposed RL-SJF algorithm, respectively

Table 4. Status of tasks computed by RL-SJF algorithm across all scenarios.

	RL-SJF														
	Number of Virtual Machines (VMs)														
Task Fault	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**100**	**200**	**300**	**400**	**500**
VMs unavailable	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Security Breach	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Deadlocked VMs	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Service Denied	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Loss of Data	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Hijacked Accounts	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Violations of SLAs	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Insufficient RAM	×	√	√	√	√	√	√	√	√	√	√	√	√	√	√
No Fault	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

From Table 3, we can observe that the cloud faces damage when it processes and computes the fault-generating tasks using the existing SJF algorithm. However, from Table 4, we can observe that with the RL-SJF algorithm, the cloud handles all the problematic fault-generating tasks and processes and computes many more tasks than the SJF algorithm. Table 4 depicts the comparative percentage of successful and failed tasks processed and computed by the SJF and RL-SJF algorithms concerning all scenarios.

From Table 5, the following observations can be made across all scenarios:

- ↑ VM = ↑ Cost: The overall cost required rises with the number of VMs.

- Average Tasks Computed Successfully (SJF) = 11.1017%.
- Average Tasks Computed Successfully (RL-SJF) = 55.5416%.
- Increase in Task Computations (In terms of folds) by RL-SJF when compared to SJF = 4.9943.
- Average Tasks Failed to Compute (SJF) = 88.8984%.
- Average Tasks Failed to Compute (RL-SJF) = 44.5585%.
- Decrease in Task Computations (in terms of folds) by RL-SJF when compared to SJF = 1.9952.
- Performance (RL-SJF) > Performance (SJF) concerning Fault-Tolerance mechanism across all the scenarios.

Table 5. Percentage of Successful and Failed Task Size processed and computed by the SJF and RL-SJF algorithm across all the scenarios.

		Computing Tasks with Respect to Success			Computing Tasks with Respect to Failure		
Scenario	**VMs**	**SJF (in %)**	**RL-SJF (in %)**	**Improvement in Terms of Success Computations (In Terms of Folds)**	**SJF**	**RL-SJF**	**Improvement in Terms of Failed Computations (In Terms of Folds)**
1	5	10.9771	55.5819	5.0635	89.0230	44.4182	2.0043
2	10	11.1475	55.552	4.9834	88.8526	44.4481	1.9991
3	15	11.1537	55.6627	4.9906	88.8464	44.3374	2.0039
4	20	11.1388	55.4214	4.9756	88.8613	44.5787	1.9934
5	25	11.0654	55.7399	5.0374	88.9347	44.2602	2.0094
6	30	11.0530	55.4152	5.0136	88.9471	44.5849	1.9951
7	35	11.3615	55.5458	4.8890	88.6386	44.4543	1.9940
8	40	11.0505	55.6466	5.0357	88.9496	44.3535	2.0055
9	45	10.9435	55.3356	5.0565	89.0566	44.6645	1.9940
10	50	11.1226	55.2298	4.9656	88.8775	44.7703	1.9852
11	100	11.1021	55.3559	4.9861	88.898	44.6442	1.9913
12	200	11.1022	55.3273	4.9835	88.8979	44.6728	1.9900
13	300	11.1023	55.2987	4.9808	88.8978	44.7014	1.9887
14	400	11.1025	55.2702	4.9782	88.8976	44.7299	1.9874
15	500	11.1026	55.2416	4.9756	88.8975	44.7585	1.9862
Average		11.1017	55.4416	4.9943	88.8984	44.5585	1.9952

5. Validating Experimental Results Using Empirical Analysis

This section includes the validation and verification of the experimental results concerning the resource-scheduling process and fault-tolerance mechanism by performing an empirical analysis using the mathematical model of R^2 analysis across all the scenarios. The parameters considered for the same are:

- Linear regression equation: represents the linear relationship between the computational cost required by the SJF and RL-SJF algorithms against the number of VMs in all the scenarios.
- Regression line slope: represents the change in the computational cost required by the SJF and RL-SJF algorithms for one-unit change in the number of VMs across all the scenarios.
- Slope sign: represents if the slope is either positive or negative.
- Y-intercept of line: depicts a point where the regression line crosses the required cost by the VM.
- Relationship (positive/negative): positive relationship indicates that the cost required for computations increases with an increase in the number of VMs; negative relationship indicates that the cost required for computations decreases with a decrease in the number of VMs.

- R^2: represents a statistical measurement representing the variance proportion of how well the regression line fits the data points in the graph of cost required against the number of VMs in each scenario.

The empirical analysis is presented into two sub-sections: Section 5.1 includes the empirical analysis with respect to the resource-scheduling process; Section 5.2 includes the empirical analysis with respect to the fault-tolerance mechanism.

5.1. Empirical Analysis Concerning Resource Scheduling

This Section 5.1 includes the empirical analysis concerning the resource-scheduling process across all the scenarios. Figure 3 depicts the cost comparison graph of SJF and RL-SJF algorithms across all the ten scenarios.

From Figure 4, we can observe that the cost required by the proposed RL-SJF algorithm is less than that of SJF across all the scenarios. Table 6 represents the empirical analysis of SJF and RL-SJF concerning the cost required across all the scenarios.

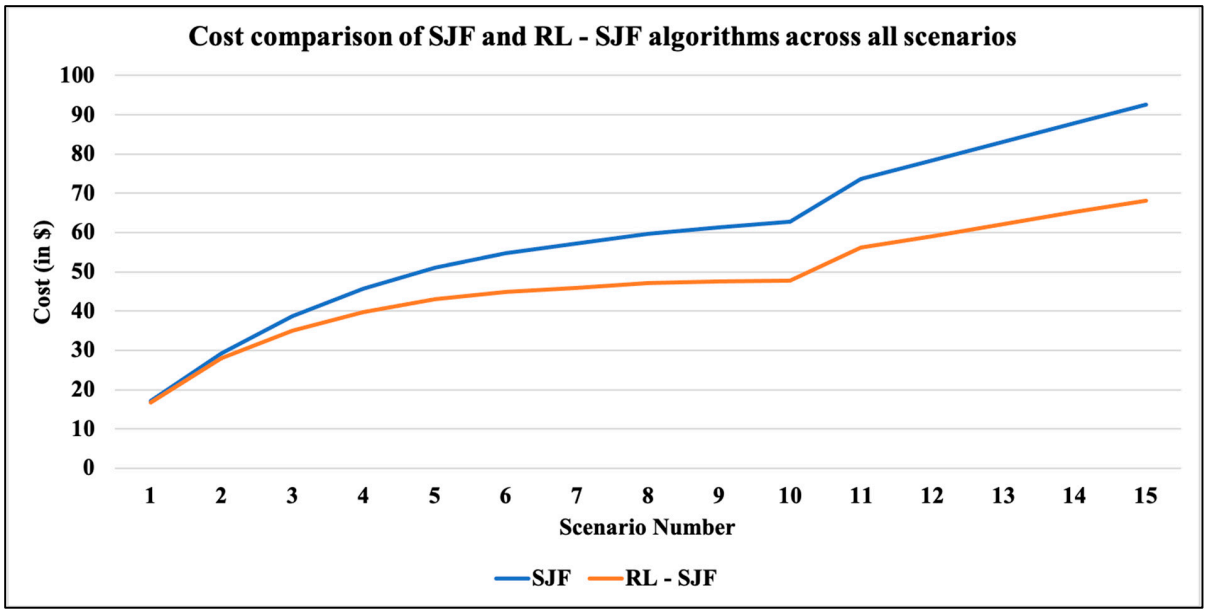

Figure 4. Cost comparison of SJF and RL-SJF algorithms across all scenarios.

Table 6. Empirical analysis of SJF and RL-SJF concerning cost required.

Parameters	SJF	RL-SJF
Linear Regression Equation	y = 4.7213x + 21.771	y = 3.0036x + 23.078
Regression Line Slope	4.7213	3.0036
Slope Sign	Positive	Positive
Y-Intercept of Line	21.771	23.078
Relationship	Positive	Positive
R^2	0.9640	0.9279
Analysis of VMs	↑ VM = ↑ Cost	↑ VM = ↑ Cost
Overall Performance	RL-SJF > SJF	

From Table 6, the following observations can be made across all scenarios:

- ↑ VM = ↑ Cost: The overall cost required rises with the number of VMs.
- R^2 (SJF) = 0.9640.
- R^2 (RL-SJF) = 0.9279.
- R^2 (RL-SJF) < R^2 (SJF): The lower value of R^2 for RL-SJF indicates that the cost required by RL-SJF is lower than the SJF algorithm.
- Performance (RL-SJF) > Performance (SJF)

5.2. Empirical Analysis Concerning Fault-Tolerance

This sub-section includes the empirical analysis concerning the Fault-Tolerance mechanism across all the scenarios. Figure 4 depicts the graph of successful and failed tasks in terms of percentage for SJF and RL-SJF algorithms across all the scenarios.

From Figure 5, we can observe that the RL-SJF algorithm managed to compute a greater number of the tasks than the SJF algorithm across all the scenarios. Table 7 represents the empirical analysis of SJF and RL-SJF concerning the fault-tolerance mechanism across all the scenarios.

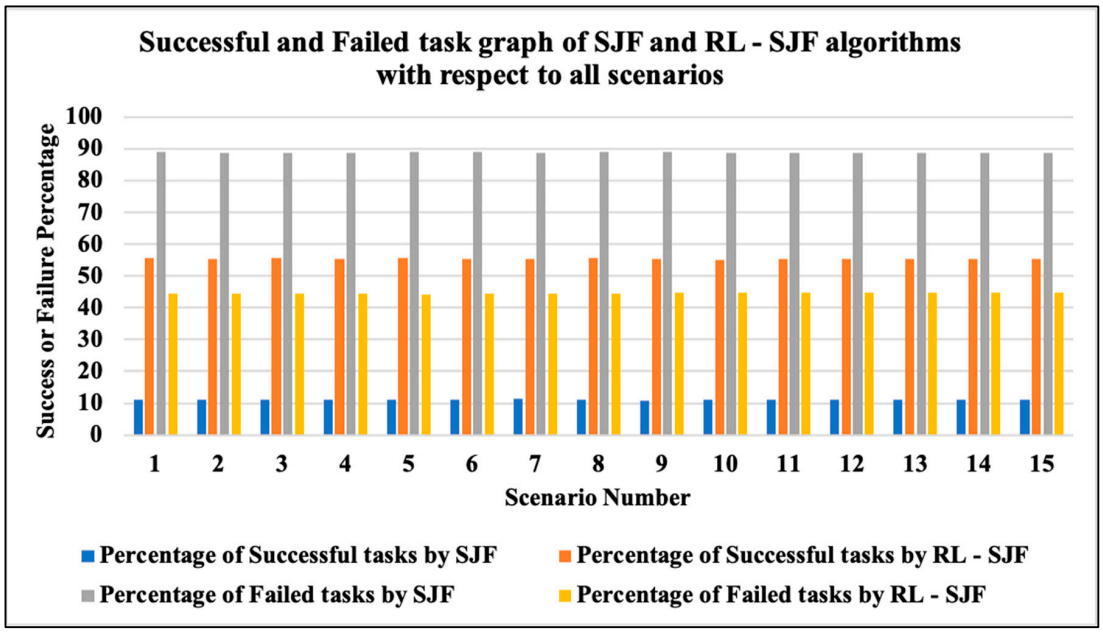

Figure 5. Successful and failed task graph for SJF and RL-SJF across all the scenarios.

Table 7. Empirical analysis of SJF and RL-SJF concerning fault-tolerance mechanism.

	Tasks Successfully Computed		**Tasks Failed to Compute**	
Parameters	**SJF**	**RL-SJF**	**SJF**	**RL-SJF**
Linear Regression Equation	y = 0.0001x + 11.101	y = −0.0286x + 55.67	y = −0.0001x + 88.899	y = 0.0286x + 44.33
Regression Line Slope	0.0001	−0.0286	−0.0001	0.0286
Slope Sign	Positive	Negative	Negative	Positive
Y-Intercept of Line	11.101	55.67	88.899	44.33
Relationship	Positive	Negative	Negative	Positive
R^2	1×10^{-5}	0.2946	1×10^{-5}	$R^2 = 0.2946$
Overall Performance	RL-SJF > SJF		RL-SJF > SJF	

From Table 7, the following observations can be made across all scenarios:

- R^2 (SJF) = 1×10^{-5} with respect to successfully computed tasks.
- R^2 (RL-SJF) = 0.2946 with respect to successfully computed tasks.
- R^2 (RL-SJF) > R^2 (SJF): The higher value of R^2 for RL-SJF indicates that the RL-SJF algorithm managed a better fault-tolerance mechanism with respect to successfully computing tasks.
- R^2 (SJF) = 1×10^{-5} with respect to failed computed tasks.
- R^2 (RL-SJF) = 0.2946 with respect to failed computed tasks.
- R^2 (RL-SJF) > R^2 (SJF): The higher value of R^2 for RL-SJF indicates that the RL-SJF algorithm managed better fault-tolerance mechanism with respect to failure of computing tasks.
- Performance (RL-SJF) > Performance (SJF) concerning fault-tolerance.

6. Conclusions

Resource-scheduling algorithms are vital for the cloud-computing environment to provide consistent and best results to the end user. Currently, without any intelligence mechanism, these algorithms perform inappropriate resource scheduling leading to limited results from the cloud. Additionally, on several occasions, the tasks being computed on the cloud VMs generate dynamic faults, thereby hampering the cloud's overall throughput and performance. To solve these issues, this research paper proposed an RL–SJF algorithm to improve the resource-scheduling process and provide a fault-tolerance mechanism at the cloud end. From the experiment and the results, the RL-SJF enhances the resource-scheduling process by lowering the required cost of computing tasks by 14.88% when compared with the SJF algorithm. Additionally, from the total tasks to be computed, the RL-SJF algorithm successfully processed and computed 55.52% of the overall tasks compared to 11.11% of the SJF algorithm. To verify and validate these experimental results, an additional extensive empirical analysis is also performed on these results. Thus, the RL-SJF algorithm provides much-needed intelligence to the cloud and improves its overall performance.

Author Contributions: Conceptualization, P.K. and J.S.; Methodology, P.L., P.K. and J.S.; Software, P.K.; Validation, P.K.; Formal analysis P.L.; Investigation, P.K.; Resources J.S.; Data curation P.L.; Writing—original draft P.L.; Writing—review & editing J.S.; Visualization J.S.; Supervision, P.K. and J.S.; Project administration P.K.; Funding acquisition, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
2. Dillon, T.S.; Wu, C.; Chang, E. Cloud Computing: Issues and Challenges. In Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, Australia, 20–23 April 2010.
3. Prasad, M.R.; Naik, R.L.; Bapuji, V. Cloud Computing: Research Issues and Implications. *Int. J. Cloud Comput. Serv. Sci.* **2013**, *2*, 134–140. [CrossRef]
4. Kumari, P.; Kaur, P. A survey of fault tolerance in cloud computing. *J. King Saud Univ.—Comput. Inf. Sci.* **2021**, *33*, 1159–1176. [CrossRef]
5. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2015.
6. Cui, T.; Yang, R.; Wang, X.; Yu, S. Deep Reinforcement Learning-Based Resource Allocation for Content Distribution in IoT-Edge-Cloud Computing Environments. *Symmetry* **2023**, *15*, 217. [CrossRef]
7. Akhtar, M.; Hamid, B.; Ur-Rehman, I.; Humayun, M.; Hamayun, M.; Khurshid, H. An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling. *J. Appl. Environ. Biol. Sci.* **2015**, *5*, 42–46.
8. Vivekanandan, D.; Wirth, S.; Karlbauer, P.; Klarmann, N. A Reinforcement Learning Approach for Scheduling Problems with Improved Generalization through Order Swapping. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 418–430. [CrossRef]
9. Yang, H.; Ding, W.; Min, Q.; Dai, Z.; Jiang, Q.; Gu, C. A Meta Reinforcement Learning-Based Task Offloading Strategy for IoT Devices in an Edge Cloud Computing Environment. *Appl. Sci.* **2023**, *13*, 5412. [CrossRef]
10. Aberkane, S.; Elarbi-Boudihir, M. Deep Reinforcement Learning-based anomaly detection for Video Surveillance. *Informatica* **2022**, *46*, 131–149. [CrossRef]
11. Sheng, S.; Chen, P.; Chen, Z.; Wu, L.; Yao, Y. Deep Reinforcement Learning-Based Task Scheduling in IoT Edge Computing. *Sensors* **2021**, *21*, 1666. [CrossRef]
12. Shin, D.; Kim, J. Deep Reinforcement Learning-Based Network Routing Technology for Data Recovery in Exa-Scale Cloud Distributed Clustering Systems. *Appl. Sci.* **2021**, *11*, 8727. [CrossRef]
13. Zhang, Z.; Liu, H.; Zhou, M.; Wang, J. Solving Dynamic Traveling Salesman Problems with Deep Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 2119–2132. [CrossRef]

14. Afshar, R.; Zhang, Y.; Firat, M.; Kaymak, U. A State Aggregation Approach for Solving Knapsack Problem with Deep Reinforcement Learning. In Proceedings of the Asian Conference on Machine Learning, Bangkok, Thailand, 18–20 November 2020; pp. 81–96.
15. Cappart, Q.; Moisan, T.; Rousseau, L.; Prémont-Schwarz, I.; Cire, A.A. Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization. *Proc. Conf. AAAI Artif. Intell.* **2021**, *35*, 3677–3687. [CrossRef]
16. Chien, W.; Weng, H.J.; Lai, C. Q-learning based collaborative cache allocation in mobile edge computing. *Future Gener. Comput. Syst.* **2020**, *102*, 603–610. [CrossRef]
17. Li, Y.; Fadda, E.; Manerba, D.; Tadei, R.; Terzo, O. Reinforcement Learning Algorithms for Online Single-Machine Scheduling. In Proceedings of the Computer Science and Information Systems (FedCSIS), Leipzig, Germany, 1–4 September 2019.
18. Liu, C.; Chang, C.F.; Tseng, C. Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems. *IEEE Access* **2020**, *8*, 71752–71762. [CrossRef]
19. Wang, X.; Wang, C.; Li, X.; Leung, V.C.M.; Taleb, T. Federated Deep Reinforcement Learning for Internet of Things With Decentralized Cooperative Edge Caching. *IEEE Internet Things J.* **2020**, *7*, 9441–9455. [CrossRef]
20. Zhang, C.; Song, W.Y.; Cao, Z.; Zhang, J.; Tan, P.H.; Chi, X. Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1621–1632.
21. Ren, J.; He, Y.; Yu, G.; Li, G.Y. Joint Communication and Computation Resource Allocation for Cloud-Edge Collaborative System. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference, Marrakesh, Morocco, 15–18 April 2019.
22. Yuan, W.; Yang, M.; He, Y.; Wang, C.; Wang, B. Multi-Reward Architecture based Reinforcement Learning for Highway Driving Policies. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019.
23. Li, J.; Hui, P.; Lv, T.; Lu, Y. Deep reinforcement learning based computation offloading and resource allocation for MEC. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018.
24. Sun, Y.; Peng, M.; Mao, S. Deep Reinforcement Learning Based Mode Selection and Resource Management for Green Fog Radio Access Networks. *IEEE Internet Things J.* **2018**, *6*, 1960–1971. [CrossRef]
25. Zhang, C.; Liu, Z.; Gu, B.; Yamori, K.; Tanaka, Y. A Deep Reinforcement Learning Based Approach for Cost- and Energy-Aware Multi-Flow Mobile Data Offloading. *IEICE Trans. Commun.* **2018**, *101*, 1625–1634. [CrossRef]
26. Zhong, C.; Gursoy, M.C.; Velipasalar, S. A deep reinforcement learning-based framework for content caching. In Proceedings of the 2018 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 21–23 March 2018.
27. Kolomvatsos, K.; Anagnostopoulos, C. Reinforcement Learning for Predictive Analytics in Smart Cities. *Informatics* **2017**, *4*, 16. [CrossRef]
28. Hussin, M.; Asilah Wati Abdul Hamid, N.; Kasmiran, K.A. Improving reliability in resource management through adaptive reinforcement learning for distributed systems. *J. Parallel Distrib. Comput.* **2015**, *75*, 93–100. [CrossRef]
29. Gabel, T.; Riedmiller, M. Distributed policy search reinforcement learning for job-shop scheduling tasks. *Int. J. Prod. Res.* **2011**, *50*, 41–61. [CrossRef]
30. Vengerov, D. A reinforcement learning approach to dynamic resource allocation. *Eng. Appl. Artif. Intell.* **2007**, *20*, 383–390.
31. Chen, W.; Liu, Y.; Dai, Y.; Luo, Z. Optimal Qos-Aware Network Slicing for Service-Oriented Networks with Flexible Routing. In Proceedings of the ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022.
32. Poryazov, S.; Saranova, E.; Andonov, V. Overall Model Normalization towards Adequate Prediction and Presentation of QoE in Overall Telecommunication Systems. In Proceedings of the 2019 14th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS), Niš, Serbia, 23–25 October 2019.
33. Manzanares-Lopez, P.; Malgosa-Sanahuja, J.; Muñoz-Gea, J.P. A Software-Defined Networking Framework to Provide Dynamic QoS Management in IEEE 802.11 Networks. *Sensors* **2018**, *18*, 2247. [PubMed]
34. Chen, W.; Deelman, E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In Proceedings of the 2012 IEEE 8th International Conference on E-Science, Chicago, IL, USA, 8–12 October 2012.