systems

MDPI

Article

A Proposal for a Tokenized Intelligent System: A Prediction for an AI-Based Scheduling, Secured Using Blockchain

Osama Younis [1,*], Kamal Jambi [1], Fathy Eassa [1] and Lamiaa Elrefaei [2]

[1] Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; kjambi@kau.edu.sa (K.J.); feassa@kau.edu.sa (F.E.)
[2] Department of Electrical Engineering, Faculty of Engineering at Shoubra, Benha University, Cairo 11672, Egypt; lamia.alrefaai@feng.bu.edu.eg
* Correspondence: oabdelhamied0001@stu.kau.edu.sa

Abstract: Intelligent systems are being proposed every day as advances in cloud systems are increasing. Mostly, the services offered by these cloud systems are dependent only on their providers, without the inclusion of services from other providers, specialized third parties, or individuals. This 'vendor lock-in' issue and the limitations related to offering tailored services could be resolved by allowing multiple providers or individuals to collaborate through intelligent task scheduling. To address such real-world systems' limitations in provisioning and executing heterogeneous services, we employed Blockchain and Deep Reinforcement Learning here; the first is used for the token-based secured communication between parties, and the latter is to predict the appropriate task scheduling; hence, we guarantee the quality of not only the immediate decision but also the long-term. The empirical results show a high reward achieved, meaning that it accurately selected the candidates and adaptably assigned the tasks based on job nature and executors' individual computing capabilities, with 95 s less than the baseline in job completion time to maintain the Quality of Service. The successful collaboration between parties in this tokenized system while securing transactions through Blockchain and predicting the right scheduling of tasks makes it a promising intelligent system for advanced use cases.

Keywords: intelligent systems; artificial intelligence; blockchain; software architecture; cloud computing systems

Citation: Younis, O.; Jambi, K.; Eassa, F.; Elrefaei, L. A Proposal for a Tokenized Intelligent System: A Prediction for an AI-Based Scheduling, Secured Using Blockchain. *Systems* **2024**, *12*, 84. https://doi.org/10.3390/systems12030084

Academic Editors: Nuno Lopes and Joaquim Gonçalves

Received: 10 January 2024
Revised: 19 February 2024
Accepted: 3 March 2024
Published: 6 March 2024

1. Introduction

We are currently depending on cloud computing systems for our everyday tasks, whether it is for work, personal, commerce, financial transactions, or data storage tasks, to name a few. Cloud services can be offered through a single provider or multiple clouds federated as a single one with a variety of services, called federated cloud, which is the interconnection and cooperation of multiple cloud computing environments [1]. The IEEE 2302–2021 Standard for Intercloud Interoperability and Federation (SIIF) highlights the importance of establishing common mechanisms and APIs to enable resource sharing among different stakeholders, fostering the testing of complex technologies in a federated environment [2]. The need for more secured heterogeneous cloud services from federated clouds and the demand for high efficiency in performing tasks on multiple clouds are increasing nowadays, especially in cases that require high confidentiality and reliability [3–6]. As known, there are potential risks associated with cloud computing, like exposure or leakage of data, unauthorized access to data, insider threats, regulatory violations, and others [7–10]. Also, there are additional challenges that need to be managed to make it more suitable for advanced use cases. For example, sensitive data or computing tasks may require a careful selection of cloud resources to process these tasks securely and efficiently, or they might need specialized software or hardware to properly execute these

tasks. Therefore, to minimize the centralization issue of cloud platforms on their respective providers, we can benefit from the evolution of web 3.0, which allows us to utilize the capabilities of emerging technologies to overcome these challenges to some extent and achieve the desired enhancements. One of these technologies is Blockchain technology, with its unique features that include immutability, decentralization, enhanced security, and faster settlement through its smart contract system [11].

Blockchain is like a distributed state machine that operates over a consensus, it is a place in which to store information, and we do not need central authorities or external parties to ensure the integrity of that data. It is considered as a database that operates over different kinds of technologies that allow you to store and to process data without the intervention of third parties, where you can just trust the network itself that is going to be completing the job [12]. Blockchain is suitable for handling reliable interactions and secured sharing of messages between parties in the Blockchain. This distributed system is immutable, so all transactions of all parties are recorded in a reliable way, either in a permissioned or permissionless Blockchain. Figure 1 shows the Blockchain architecture.

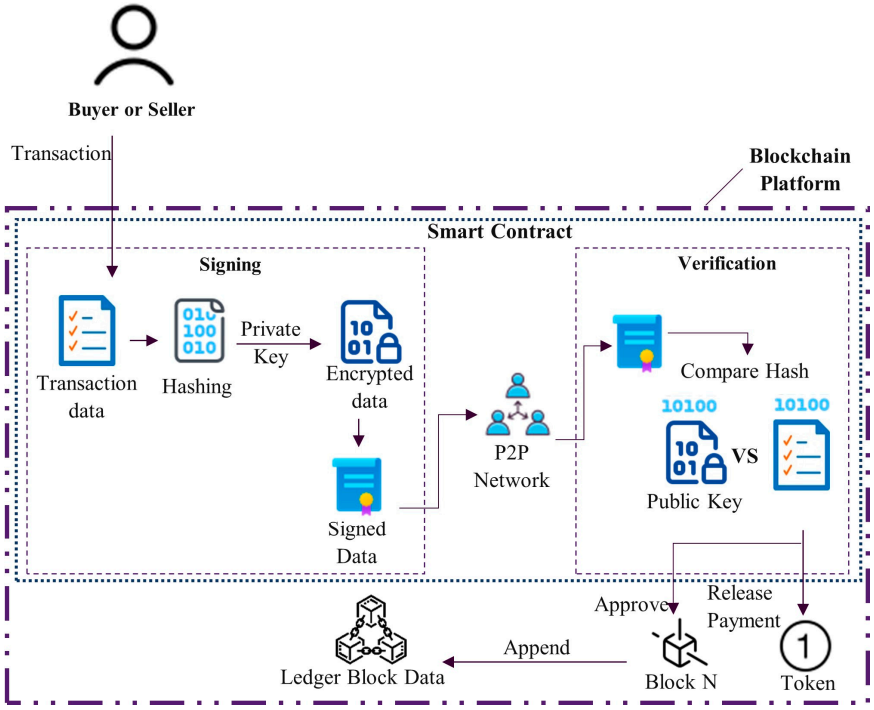

Figure 1. Blockchain architecture.

Blockchain-enabled systems provide more enhanced capabilities as compared to conventional cloud-based systems, as shown in Figure 2, and allow for open-ended opportunities and capabilities for Society 5.0 and Industry 4.0 [13].

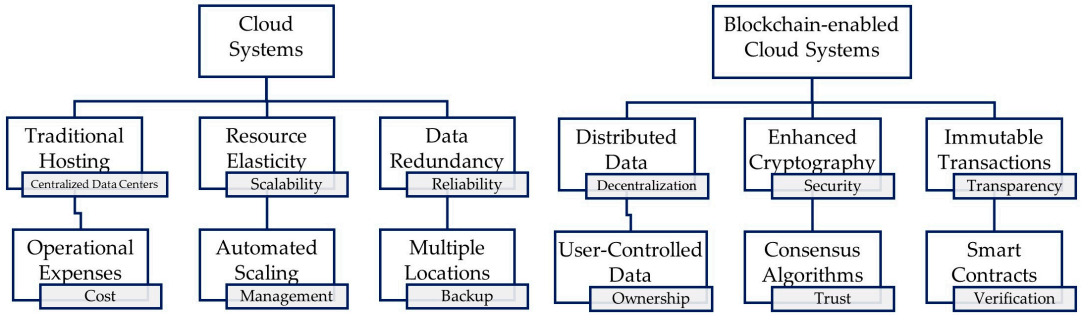

Figure 2. Conventional cloud systems vs. Blockchain-enabled cloud systems.

One use case of utilizing such capabilities, which is the proposed system here in this paper, is to split a job into multiple tasks or chunks to be processed separately through multiple different nodes "executors" that are connected though a Blockchain network to execute the required tasks without involving any intermediary, making use of its smart contracts to guarantee a quick and secured settlement and transparent task execution. We employed Blockchain technology here to immutably store all transactions to avoid the possibility of data manipulation or the violation of Quality of Service (QoS), also to link the service requester to the distributed executors, whether they are cloud resources or individual service providers. This way we can achieve confidentiality through executing fragmented sensitive tasks, perform compute-intensive tasks that are not possible using a single machine, and execute tailored services that require specialized hardware or software, all without the constraints of a central provider. In this case, the individual computation capability of each executor impacts the overall execution efficiency, which will be considered before being selected to execute the required job. Also, there should be a mechanism of reputation to be considered before selecting an executor to ensure execution trustworthiness, instead of randomly choosing executors or choosing them only based on stakes. It is known that task scheduling affects the efficiency of execution, so designing their schemes properly is an important challenge. Therefore, we want to design a mechanism that can address the aforementioned issues in an efficient way.

It is difficult to use traditional optimization mechanisms that are based on static optimization, which may not be suitable for the dynamic and complex features of distributed environments such as cloud computing [14]. Optimizing the performance on the long term requires designing an efficient policy that considers the challenges related to the dynamicity of the system, like workload nature, QoS requirements, and the characteristics of candidates (potential executors), which usually have an enormous state space. To tackle the issue of dynamicity and complexity of the federated cloud environment, we can involve artificial intelligence (AI) here to further enhance the proposed system. Nowadays, AI is being utilized in many different fields, with endless applications including prediction, Internet-of-Things (IoT), classification, detection, recognition, etc. [15–20]. The AI technique that we employed here is Deep Reinforcement Learning (DRL), a promising approach in handling advanced control problems [21–23]. It comprises Reinforcement Learning (RL) and Deep Learning (DL), where RL learns to make the right decisions through taking the best-known actions then, based on the action taken, receives penalties or rewards as feedback. Whereas, DL approximates the complex functions where deep neural networks learn to represent the value of actions in different states to correlate state and action that are correspondent with the value function. The incorporation of such techniques is also called hyper-heuristics, which are high-level search methodologies [24–26].

The decisions in DRL are made under the guidance of deep neural network (DNN), which provides precise prediction and estimation that lead to a long-term optimal solution. As it is not an easy-to-handle problem, we employed the Deep Q-Learning (DQL) technique, which is a specific algorithm within DRL that specifically addresses the problem of learning the action value for a given state to tackle the challenges of selecting the proper executor and resource allocation, which helps in reducing the huge size of the action space. This significant reduction resulted from applying Deep Q-Network (DQN) that eliminates the need for a table to store Q-values for each state–action pair; instead, it utilizes the DNN to estimate these Q-values.

The contribution of this work is to build a proof-of-concept system that allows service requestors to benefit from broad range of services offered heterogeneously by cloud providers and, later, individuals in a decentralized collaborative way that minimizes the centralization issue of single cloud platforms. Also, it enhances the overall process security in terms of immutability and integrity of all related transactions. This incentive-based system integrates DQL and Blockchain to predict the appropriate task scheduling for executors that are available on federated clouds, avoiding potential vulnerabilities and taking advantage of the smart contracts feature. This proof-of-concept system is validated on a

federated cloud environment and successfully performs a job through choosing suitable executors for that job by considering their computing capabilities and job type. This system is applicable in different fields including, but not limited to, trading systems, E-commerce, E-government, open-source intelligence (OSINT) and others.

The rest of the paper is structured as follows: Section 2 summarizes the literature review, Section 3 describes the proposed system, Section 4 discusses the results of the implementation, and finally, Section 5 concludes the paper and briefly mentions the planned future work related to this topic.

2. Literature Review

Blockchain technology has recently gained significant attention, as it fits in almost everywhere while still providing its distinct features. While investigating the related work for this research, we found that most of the recent studies that combine Blockchain with DRL for cloud systems were mostly applied in the two fields of Internet-of-Things (IoT) and Industrial IoT [13,16,27–30]. On the other hand, only a few studies have been completed combining Blockchain and DRL in the field of token-based task scheduling prediction on federated cloud systems and for participants' evaluation, which is our focus in this work. Hence, we are highlighting some of this partially-completed related work that inspired us to work on this proof-of-concept.

Xiao, H et al. [31] have proposed a decentralized architecture to develop a Blockchain-supported Internet of Vehicles (IoV) system using an algorithm that is based on Deep Reinforcement Learning for resource optimization. They formulated a resource optimization problem and exploited the Deep Reinforcement Learning algorithm to determine the allocation scheme.

Gao, S et al. [32] have also built a resource allocation and task offloading mechanism for a video frame resolution scaling that is based on DQL, they formulated an optimization problem of cost minimization of energy, delay, and accuracy to improve the experience quality of the user equipment; they utilized DQN for dimensionality reduction while selecting the action for offloading and resource allocation, resulting in an accelerated training speed through overcoming the sparseness of single-layer reward functions.

In the field of cloud–edge–end cooperation environments, Fang, C et al. [33] have solved the issue of the overloaded mobile traffic that is resulted from the Internet services by designing a DRL-based optimization mechanism to allocate resources in heterogeneous environments with cloud–edge–end collaboration, focused on improving the distribution of content, where they schedule the tasks of the arrived content requests according to the historical requests from users. In another study supported by edge computing, Quan, T et al. [34] have designed a seismic data query mechanism using DQN for mobile edge computing; they formulated a minimization optimization problem aiming to minimize the task delay in the long-term by considering the computing capacity constraints of edge servers, resulting in outstanding performance results.

Most of the related studies have considered their proposed ideas as optimization problems, either minimization or maximization, to minimize the cost of any aspects: delay, energy, resources, operating expenses, etc., or to maximize the reward, performance, etc. Todorović, M et al. [35] have proposed a consensus mechanism based on PoUW that assumes solving instances of optimization problems for more efficient utilization of computational resources and provision incentives for the Blockchain participants.

In [36], they designed a multi-objective mechanism to optimize the response time for task scheduling and energy consumption in an edge–cloud collaborative environment. This design is based on the A3C algorithm to propose a task scheduling policy optimization algorithm, with acceptable simulation results confirming its ability to reduce the overall response time and the calculated energy consumption of their proposed system.

For the purpose of handling the problem of the unpredictability of tasks and QoS requirements of users in Fog Computing, the authors in [37] have proposed three different DRL-based solutions, with the aim of maximizing rewards with reference to resource

utilization, designed as the Markov Decision Process, by taking into consideration the task priority, energy consumption, and latency as the QoS factors. Another effort has been made in [38] for a trusted supervision of federated learning and malicious node attacks by smart contract; they proposed the concept of a decentralized trusted computing sandbox, which is a federated learning multi-task scheduling mechanism using DRL to solve the resource scheduling optimization problem. Both of these two studies have not incorporated Blockchain in their proposed ideas of resolving the scheduling optimization problem. On the other hand, ref. [39] has proposed a DRL-aware Blockchain-based task scheduling system for healthcare applications through a solution that provides makespan efficient scheduling to solve the issues of task scheduling, security, and the cost related to processing tasks in the IIoT healthcare paradigms. We have also been inspired by the work carried out in [40], where they have introduced a multi-agent collaborative DRL-based scheduling algorithm with DQN in Mobile Edge Computing (MEC) to solve the issues of low latency, offloading, and task scheduling in MEC using Karush–Kuhn–Tucker (KKT) to minimize the total latency and DQN to reduce the energy consumption.

To the best of our knowledge, and compared to IoT-related contributions, a few studies have been carried out for enhancing the utilization of federated clouds, and none of the studies in the literature review have proposed an incentive-based system integrating the Blockchain features with DQL in federated clouds, nor have they utilized the DRL capabilities to select nodes (executors) reliably by predicting efficient task scheduling. In addition, the effectiveness of Blockchain-enabled cooperative execution is not yet studied thoroughly, and there should be more evaluation mechanisms for selecting the proper distributed executors. Therefore, we are trying here to fill this research gap and propose this system as a proof-of-concept applicable in use cases that minimize cloud providers' lock-in issue by allowing for heterogeneous services from federated cloud and, later, the individuals who offer specialized token-based services.

3. Methodology

The methodology we followed here was used to build a proof-of-concept solution that integrates Blockchain technology with a DRL approach with the purpose of enabling a collaboration among the distributed executors using a rewarding mechanism as an incentive, also for the purpose of creating an evaluation mechanism for selecting the proper executors. This methodology should allow for secured transactions on federated clouds by taking advantage of the immutable Blockchain and the smart contract features, while also utilizing the DRL approach to predict appropriate task scheduling.

The notations used in this section are tabulated in Table 1 below, with a description of each notation to help readers easily understand the abbreviations and algorithms.

Table 1. Notations.

Symbol	Description
FCM	Federated Cloud Manager
Predictor	Deep Reinforcement Learning agent
BCM	Blockchain manager
RC	Resource Collector
N	Number of tasks
$\mathcal{T}$	Timeout, the assumed QoS
Q	Reward (token per task)
Job	N, D, and Q
n^*	Scheduling profile
f^*	Computational resource allocation profile
W^*	Task scheduling and resource allocation policy
$\boldsymbol{\theta}_C$	Candidates set; the potential executors not yet selected
$\boldsymbol{\theta}_E$	Executors set; the selected executors from the candidates set

In the following subsections, we explain the workflow and the design of the proposed concept and its architecture, in addition to how we defined the state space, action space, and reward function.

3.1. Workflow of the Proposed System

The *requester* submits a job request with the dedicated reward and the number of tokens Q needed to pay for the successfully executed job. This request transaction includes the workload (number of tasks N), the QoS requirement (assumed as timeout $\mathcal{T}$), and the reward Q. To handle these transactions, as we adopted DPoS here, the block creators (elected by the stakeholders) will generate a block and append it to the Blockchain; then, after reaching a consensus, they will add the produced block permanently to the Blockchain. This job request transaction is broadcasted through the BCM and received by the Predictor, the DQL agent, which will assess the job type and perform the algorithm to select the suitable executor set θ_E from the already available candidate set θ_C. This selection is carried out through the evaluation process of the candidates, which evaluates the S, V, and F values of each candidate. Ideally, any service providers or nodes on the network who have submitted an execution availability transaction can be considered as candidates, but here, the Predictor collects and groups all resources (including VMs as executors) that are available in the federated cloud into a candidate set θ_C.

After candidate evaluation and executor selection, and based on the selected executors, the Predictor will design the appropriate policy of task scheduling on the selected executors and send it to the *requester*; the result of this selection is also appended to the Blockchain. Considering that the selected executors in θ_E are VM instances, the FCM will initiate and initialize the designated instances to be ready by uploading the related folders with scripts to execute the workload. The *requester* should then dispatch the required job tasks to the executor and send the output IDs to the Blockchain to announce job acceptance and the executor acknowledgment for the job approval, which, in turn, will execute the job based on the specified policy.

For the executors, to obtain the highest reward, they should allocate the related computational resources to successfully execute the job and meet the QoS requirements efficiently. After successful job execution, they send a completion announcement on the Blockchain when it is completed. This process will either return true if successful (no violations of QoS) or false if otherwise. Finally, and only if successful, the smart contract will release the allocated tokens, and the executor will obtain the reward.

All these transactions, along with the token ledger, are secured by and immutably recorded on the Blockchain, guaranteeing the transparency and integrity of transactions for both the *requester* and executor, where all parties are connected through the Blockchain network. Figure 3 illustrates how different parties are connected in the proposed concept.

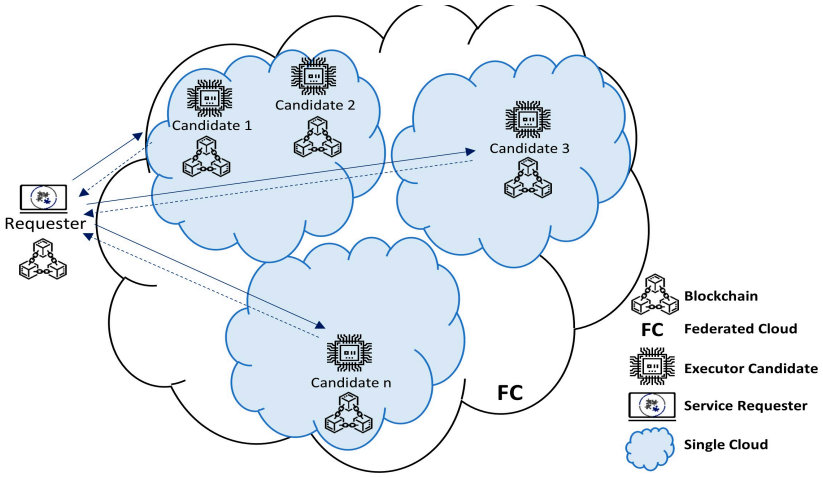

Figure 3. Illustration of how parties are connected in the proposed system.

The workflow of the proposed system is formulated in Algorithm 1 below, whereas Algorithm 2 explains the steps taken by the Predictor (Deep Reinforcement Learning agent) to select the appropriate executors and build task scheduling and resource allocation policy by solving problem $P1$, as shown in (9). These steps utilize a DQL (Deep Q-Network) that employs a neural network with a pre-trained main Q-Network to predict the Q-value for a given state, through two phases:

1. Estimating the "state-action" value function that is correspondent to an action–value function (offline DNN construction, steps 4 to 6 in Algorithm 2);
2. Action selection and dynamic network updating (online dynamic Deep Q-Learning, steps from 7 to 23 in Algorithm 2).

The "offline" phase here refers to setting up the DNN before the interaction with the environment, where we decide the architecture in terms of the number of layers, the configured neurons in each layer, and the applied activation functions. First, the initialization of the *Main Q-Network* weights with small random values to break the symmetry and allow for more proper learning. Then, the initialized *Main Q-Network* is duplicated by creating a *Target Q-Network*, ensuring both have the same initial weights. The use of two networks reduces the correlations between the Q-values being predicted and the targets used for training to stabilize learning. The *Experience Replay* is also initialized in this phase, which will store the Predictor's experiences in terms of state, action, reward, and next state tuples. Lastly, in this phase, we pre-trained the network using the Azure 2019 workload dataset to speed up learning.

The "online" phase involves the actual learning process, where the Predictor interacts with the environment and updates the DNN based on its experiences. Using an ε-greedy strategy, the Predictor decides whether to take a random action (explore) or the action suggested by the current policy (exploit). For each action taken, the agent receives a new state and a reward from the environment, and the tuple (current state, action, reward, next state) is stored in the *Experience Replay*. Periodically, a batch of experiences from *Experience Replay* are sampled to allow the DNN to learn from a more diverse set of experiences, reducing the correlation between consecutive learning updates and improving stability. Using sampled experiences, the *Predictor* calculates target Q-values using the *Bellman equation*, and these targets reflect the expected future rewards. To minimize the loss between the predicted Q-values and the target Q-values, we use the loss function. The weights from the DNN are copied, after several learning updates, to the *Target Q-Network* to update the targets. These steps allow the DQN to effectively learn policies that map the states to the actions to maximize the cumulative reward, accounting for both immediate and future rewards.

Algorithm 1: Workflow

1 *Requester* initiates *Job* request to Predictor and submit Job transaction to *BCM* with amount of tokens
2 **for** each Job transaction sequence **do**
3 *BCM* publicizes Job's parameters N, $\mathcal{T}$, and Q on the Blockchain network
4 *RC* groups the available candidates from *FCM* in a candidates set ϑ_C
5 Predictor performs executor selection & evaluation and builds W^* using Algorithm 2
6 Predictor sends W^* to *Requester* and *BCM*
7 *BCM* generates and appends a block to the Blockchain using DPoS mechanism
8 *Requester* sends Job to the selected Executor and Blockchain
9 Executor executes Job according to W*
10 Executor announces job completion on the Blockchain network
11 Executor sends back the completed job to *Requester*
12 Execute smart contract and release reward tokens to *Executor* if QoS is satisfied
13 **end**

Algorithm 2: Predictor Algorithm

Input: Job submitted by *Requester*
Output: Θ_E^i and W^*
Initialize: *BCM*, *FCM*, *Predictor*, *Experience Replay*, *Main Q-Network*, *Target Q-Network*
Obtain the $Q(S, A)$ value estimates
Store $Q(S, A)$ value estimates in the *Experience Replay*
Pretrain the *Main Q-Network* with (S, A) and corresponding Q value $Q(S, A))$
for each execution sequence **do**
  for each decision epoch T^i **do**
    Select random action (Epsilon-Greedy) or the action with the highest Q value estimated by *Main Q-Network* $a^i = \max_a Q(S^i, a^i)$
    Execute action a^i to obtain Θ_E^i out of the available Θ_C in *FCM*
    for each executor in Θ_E^i **do**
      Select optimal n^* and f^* using Equation (9)
      append n^* and f^* to $W^* = \{n^*, f^*\}$
    end
    Calculate the immediate reward R^i
    The chosen executors execute the job according to W in the action space A^i
    State transition observation T^{i+1} (next decision epoch) with the new state S^{i+1}
    Store this experience (S^i, A^i, R^i, S^{i+1}) into the *Experience Replay*
    Pick samples randomly (S^n, A^n, R^n, S^{n+1}) from the *Experience Replay*
    From the *Target Q-Network*, calculate target Q-Value using the equation: $y^n = R^n + \gamma \max_{a'} Q(S^{n+1}, a')$
    Use the Loss Function $L(\theta) = [y^n - Q(S^n, a'; \theta)]^2$ to update *Target Q-Network* every *C* iterations
  end
end

3.2. System Design and Architecture

We designed the architecture of this system to be based on DRL to solve the complexity of the solution and to maintain high scalability, benefitting from its distinct performance features from RL and DNN. RL adopts a Q-table to store Q-values to determine the best way to get the best results by adhering to a specific methodology; hence, it generates near-optimal control actions through immediate reward feedback by interacting with its environment [41,42]. This makes RL effective for resource allocation in dynamic systems such as the federated cloud environment. However, sometimes the decisions becom complex for the RL approach, making it harder for the Q-table to accommodate; therefore, the DNN is applied to help with estimating the potential states rather than mapping every solution to determine the reward path, which reduces the action space size. The resultant approach is DRL, which we utilized here in our system design to handle the dynamicity of tackling advanced control problems that have high-dimensional state space [43]; particularly, we adopted DQL for selecting actions and dynamically updating the network. The architecture of the proposed system is shown in Figure 4.

Some of the quality attributes we can achieve through this design are the following:

- Reliability is guaranteed through the availability of multiple (redundant) executors;
- Maintainability is achieved as we can modify to improve or adapt new consensus mechanisms, executor evaluation criteria, *Predictor*'s learning pattern enhancements, included cloud, or individual service providers, etc.;
- Interoperability is also managed as we can communicate with specific providers through their respective API, and also with participants through Blockchain;
- Security features are ensured through immutability and smart contract capabilities.

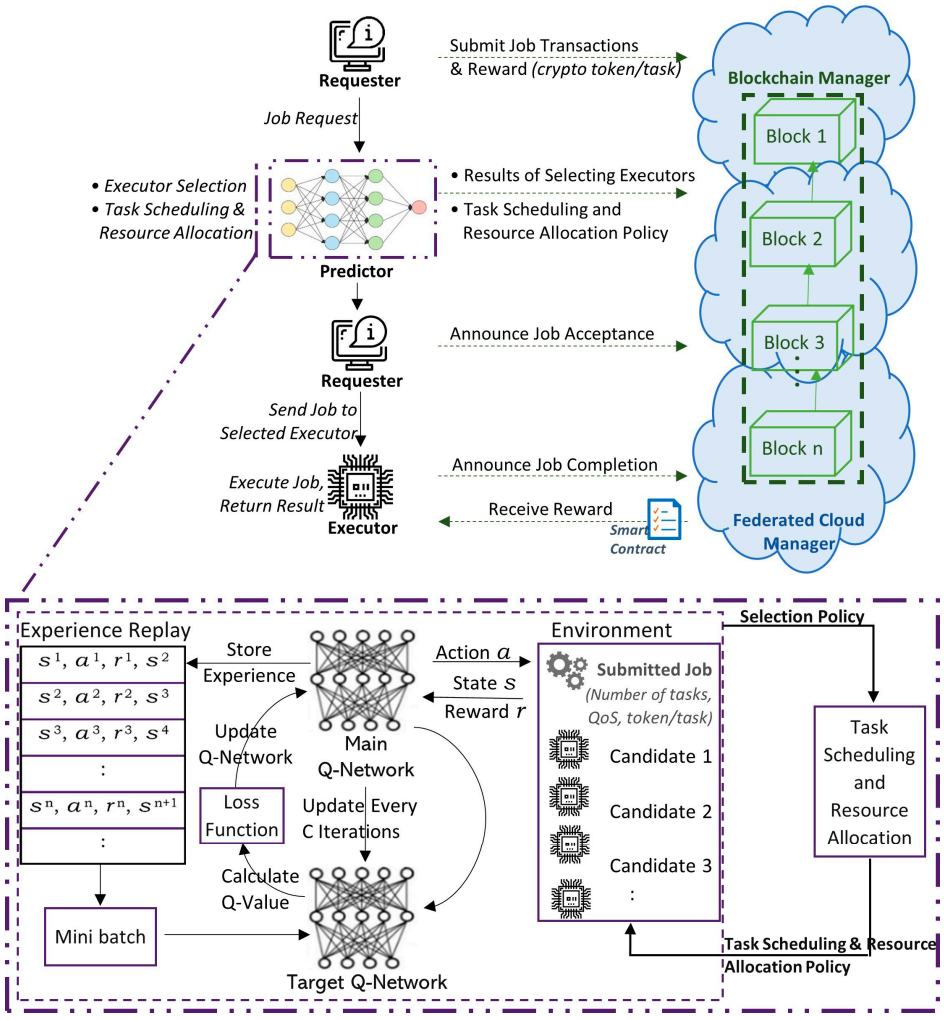

Figure 4. The proposed architecture.

The role of the task scheduling and resource allocation module, the *Predictor*, is to perform the DQL algorithms to predict the appropriate task scheduling and resource allocation policy. For that, we designed a mechanism of evaluation to assist in the process of executor selection from the available candidates by considering the computation capability of the candidates in addition to their stakes and reputation. The DQL approach is used here to deal with the complexity of the system and the huge action space related to the execution workloads, QoS requirements, and characteristics of the candidates, as explained below, which requires us to design an approach to create a manageable solution space in the decision process. To some extent, this methodology handles the enormous state space and efficiently meets the workload adaptation and QoS requirements [43,44].

The Blockchain Manger (*BCM*), on the other hand, manages all Blockchain-related transactions and appends them to the distributed ledger, including the transactions from the Predictor and both parties, the *Requestor* and the *executor*. This makes a Blockchain-enabled system that allows service providers to be the "*executors*" (for some tokens) and offers its services to the "*requesters*", who pay the tokens to avail these services or to carry out the completion of their tasks, leading to a managed token-based economy. In this system, we have adopted the Delegated Proof-of-Stake (DPoS) as the consensus mechanism, as it is flexible, efficient, and fast as compared to PoW and PoS [45]. In this consensus mechanism, stakeholders do not create the blocks themselves; they vote to elect the delegates, also referred to as witnesses or block producers, giving them the right of creating blocks; hence, there is no computational power consumption for the stakeholders.

There is also a Federated Cloud Manager (*FCM*) that manages all connected clouds in the federated cloud with their resources, of any type. It translates all actions related to resource allocation to the respective clouds, and coordinates with *BCM* for a distributed collaborative environment hosting the Blockchain ledger. The services available in the federated cloud are collected and grouped by a module called the Resource Collector (*RC*). A process diagram of these modules is simplified in Figure 5 below.

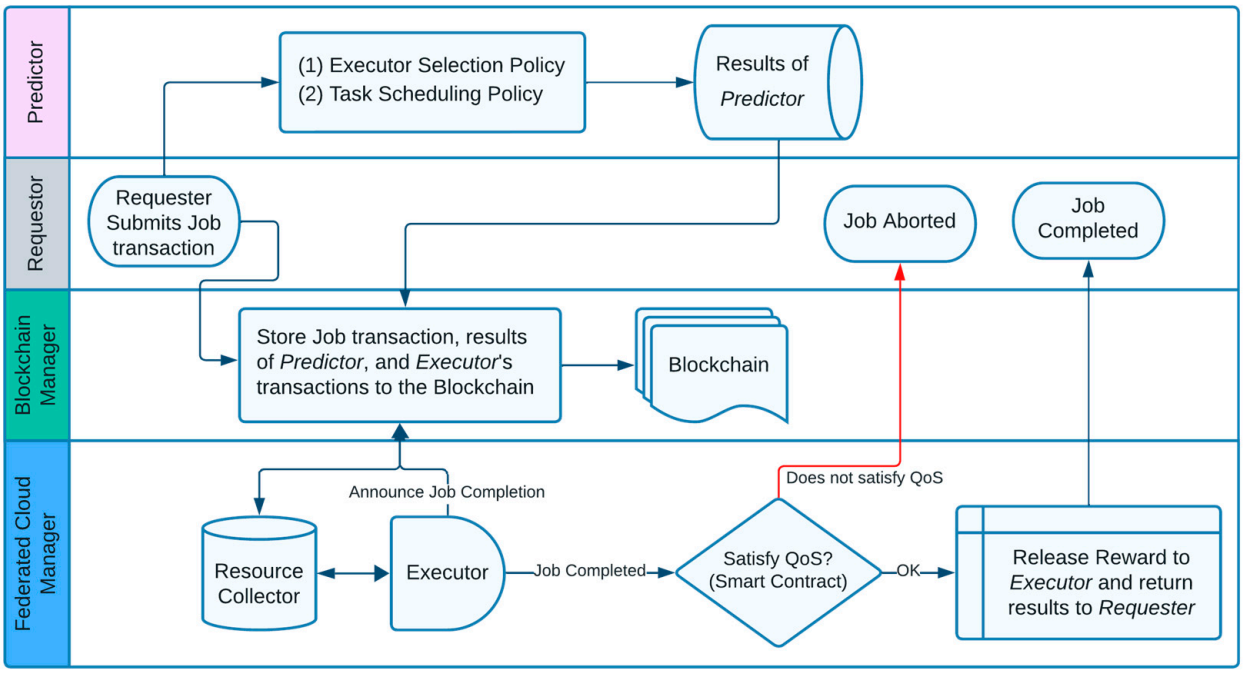

Figure 5. Process diagram of the system modules.

In the following, we define the state space of the proposed solution, how we set the action space in the simplest way, and, finally, how we formulated the reward function.

State space:

In a decision epoch T^i $(i = 1, 2, 3, \ldots)$, the state S^i is the union of the following: job workload (number of tasks N^i), QoS requirement (here, timeout $\mathcal{T}^i$), candidate's stake S_m^i, candidate's reputation value V_m^i, and computational resource F_m^i, where candidate m belongs to the *candidate set* θ_C. Hence, the state space is represented as follows:

$$S^i = \left[N^i,\ \mathcal{T}^i,\ \left\{S_m^i,\ V_m^i,\ F_m^i\right\}\right],\ U_m^C \in \theta_C \tag{1}$$

i here, and in the rest of the paper, denotes the variable in decision epoch T^i. This is not to be confused with $\mathcal{T}$, which is the timeout as the QoS requirement. For the candidate's stake, S_m^i, it comprises its own stake and the stake of others who delegate towards it; the higher the stake the better, as the candidates with higher stakes technically have more influence and a higher probability of being selected as an *executor*. For the reputation value, V_m^i, we have adopted a similar concept to the reputation-based rating mechanism proposed by Liu, G et al. [46] to determine the candidates' reputation value.

Action space:

The Predictor here should properly select a set of executors out of the candidates available, which will be carried out just after a job is released with a detailed workload. While choosing the executors, it should also design the task scheduling policy for the

selected executors, along with the resource allocation policy to execute the job successfully. Thus, we represent the action in decision epoch T^i as:

$$A^i = \left\{ a^i,\ n^i,\ f^i \right\} \tag{2}$$

Here, $a^i = \{a^i_m\},\ a^i_m \in \{0,1\},\ U^C_m \in \theta_C$, where $a^i_m = 1$ indicates that the candidate U^C_m is selected as the *executor*, and $a^i_m = 0$ indicates that candidate U^C_m is not selected as the executor. Once the action a^i is taken, the Predictor chooses the *executors set* θ^i_E from the available *candidates set* θ_C . This executors set is represented as

$$\theta^i_E = \left\{ U^C_m \middle| a^i_m = 1, U^C_m \in \theta_C \right\}, \tag{3}$$

and, therefore, θ^i_E has a cardinality of $\left|\theta^i_E\right|$ as the number of elements inside. The execution scheduling profile is

$$n^i = \left\{ n^i_j \right\}, n^i_j \in \{1, 2, \ldots, N\},\ U^c_j \in\ \theta^i_E, \tag{4}$$

and the computational resources required by the executors to perform the job are denoted as $f = \{f_m\},\ f_m \leq F_m$, where $F = \{F_m\}$ is the computational resources available, so the computational resource allocation profile is

$$f^i = \left\{ f^i_j \right\},\ 0 < f^i_j \leq F^i_j,\ U^c_j \in\ \theta^i_E \tag{5}$$

Reward function:

The reward which executors are receiving through executing the required job collaboratively is what we mean by reward here. This involves two things: the income of completing the job (which affects the *executor*'s reputation value), and the received block reward as a stakeholder (which affects the *executor*'s stake). The function can be formed as:

$$R^i\left(S^i, A^i\right) = \sum\nolimits_{U^c_j \in\ \theta^i_E} \left[S^i_j + n^i_j \left(\frac{V^i_j}{V_{max}} Q \right) \right] \tag{6}$$

where Q is the reward (token/task), wS^i_j is the weight of stake, and $\frac{V^i_j}{V_{max}}$ is the reputation value, which should be $\leq$1. But first, we must ensure that the executor has satisfied the assumed QoS requirement, the timeout; so, the executer must complete the assigned n^i_j job tasks before the timeout to receive the reward. Hence, we will consider the duration D as a constraint: $n^i_j D^i_j \leq\ \mathcal{T}^i\ \forall U^c_j \in\ \theta_E$. This constraint is calculated by:

$$n^i_j \left\{ \frac{\psi^I X}{f^i_j} \right\} \leq\ \mathcal{T}^i \forall U^c_j \in\ \theta_E, \tag{7}$$

where ψ^I represents the individual size of each job task in bytes, X is the computation intensity in CPU cycle/byte. The reward function here should represent the maximum reward achievable while also satisfying the QoS requirements. As a result, the reward function will be rewritten to be as follows:

$$R^i\left(S^i, A^i\right) = \begin{cases} \sum_{U^c_j \in\ \theta^i_E} \left[wS^i_j + n^i_j \left(\frac{V^i_j}{V_{max}} Q \right) \right], & if\ (7)\ is\ satisfied \\ 0, & if\ (7)\ is\ not\ satisfied \end{cases} \tag{8}$$

Task scheduling:

After determining the executor set θ_E that has been selected out of the candidate set θ_C, the Predictor should maximize the cumulative rewards while satisfying the constrains

related to the QoS requirements, i.e., it should build an efficient decision of task scheduling and resource allocation, taking into consideration the constraints. This decision will be considered as an optimization problem that should be solved to establish the best resource allocation strategy suitable for the *executors* selected. Usually, the optimization problem of the decision leads to a non-convex NP problem, meaning that determining the best strategy for allocating resources is difficult, especially in decentralized environments [32]. If we look at the action space A^i in (2) and the reward function $R^i\left(S^i, A^i\right)$ in (8), they form a difficult problem to solve due to the large space size and the constrained reward function. Hence, we formulated the scheduling problem into an optimization problem to satisfy the constraints and to maximize the achieved reward, Q, to the highest possible, as follows:

$$\begin{aligned} &P1: max\ Q = \sum_{U_j^c \in\ \theta_E} n_j^i \left\{ \psi^I X \left(f_j^i\right)^2 \right\} \\ &C1: \sum_{U_j^c \in\ \theta_E} n_j^i = N^i, n_j \in \left\{1, \ldots,\ N^i\right\} \\ &C2:\ n_j^i \left\{ \frac{\psi^I x}{f_j^i} \right\} \leq\ \mathcal{T}^i \forall U_j^c \in\ \theta_E \\ &C3: 0 < f_j^i \leq\ F_j^i, \forall U_j^c \in\ \theta_E \end{aligned} \tag{9}$$

The objective function is derived from (8), where constraint $C1$ is the validity of the task scheduling, as already represented in (4). $C2$ is the satisfaction of the QoS (the timeout $\mathcal{T}$), as formulated in (7). Finally, $C3$ is the computational resource allocation, as defined in (5). Since the problem includes constraints and involves randomness and uncertainty, it is computationally expensive to use traditional methods to find the best option to allocate the resources; this is because of the non-convex optimization problem [23]. For that, we employed the Deep Q-Learning approach to address this optimization problem for task scheduling and resource allocation. Deep Q-Learning was adopted to predict the workload in the federated clouds. We have built on the work carried out by Ahmed, Z. et al. [47]; they used DQL to predict the workload and demonstrated a significant improvement over popular existing scheduling methods that include Genetic Algorithm, Heteroscedastic Gaussian Processes, Median Attribute Deviation Minimum Migration Time, and Robust Logistic Regression Minimum Migration Time algorithms [47].

As part of this problem representation, we can see how our defined state space, action space, and reward function encourage the desired outcome. To maximize the reward resulting from successful task scheduling, the reward could be positive for successfully completing tasks within the QoS constraints and negative for failing to schedule a task, violating QoS constraints, or inefficient resource allocation. We incorporated logic in the environment that only allows for valid actions as part of $C1$, and implemented checks that verify if a task's scheduling and execution meet its QoS requirements to satisfy $C2$, in addition to ensuring that actions respect the computational resource constraints in terms of attempts to allocate more resources than available, which should be penalized or invalidated, to meet constraint $C3$.

We used the DNN here and initialized the Q-Network that takes the state as input and outputs the predicted Q-values for all actions in the action space, which, in turn, assesses the quality of a particular action taken in each state. This is, with the help of the initialized experience replay to store past experiences (state, action, reward, next state), used to train the Q-Network by sampling batches of experiences to reduce correlations in the observation sequence, in addition to the implementation of the Epsilon-Greedy policy for exploration and exploitation. We trained the model as described in Algorithm 2, and after sufficient training, the policy for task scheduling and resource allocation is implicitly defined by the Q-Network. For any given state, the action with the highest predicted Q-value is considered the best action under the learned policy.

As a result, the DQL can learn the optimal strategies by interacting with the environment, adapt to dynamic changes, and learn from the consequences of its actions. This solves the optimization problem $P1$ and satisfies the formulated constraints.

4. Results

We implemented and evaluated the proposed system to prove the concept and its validity. We have four main modules in this system, the Federated Cloud Manager (*FCM*), Blockchain Manager (*BCM*), *Predictor*, and Resource Collector (*RC*). The *FCM* module is responsible for managing everything related to the federated clouds. In this experiment, we chose clouds from three providers: Azure, Google Cloud Platform (GCP), and Amazon Web Services (AWS). The *BCM* module manages all transactions related to the Blockchain. Here, we chose Tron Blockchain, an open-source platform that supports decentralized applications (dApps) and enables us to write Blockchain applications in a customizable and flexible way [48]. For the test purposes, we used TronPy and the public "Testnet" network in Tron Blockchain; we built and tested all related smart contracts for validation in this public Testnet network. As mentioned earlier, we adopted the DPoS consensus mechanism in this network, as it is more scalable and needs fewer computational resources [49]. The Predictor module is where the Deep Reinforcement Learning agent acts. It performs the DQL algorithms to predict the appropriate task scheduling and resource allocation policy. Finally, the *RC* module is responsible for collecting and unifying configuration data from the three cloud infrastructure providers, Azure, GCP, and AWS. This *RC* module dynamically collects and maintains the available resources from these cloud providers (comprising the federated cloud). Here in this experiment, there are about 1200 collected resources; we categorized them by provider, machine type, memory (GB), max disk size (TB), SSD availability, number of GPUs, GPU memory, GPU model, CPUs, CPU frequency (GHz), and machine family. There are five types of machine family: memory-optimized, compute-optimized, graphic rendering, general-purpose, and high-performance compute (HPC) machine families. All of these 1200 resources are considered as the candidate set θ_C; this set is managed dynamically by *RC*, which act as a manager for all API calls.

The following subsections show the details of the experiment, the related settings, and the evaluation of the performance.

4.1. Experimental Details and Settings

Table 2 shows the configurations and parameters used in the implementation.

α controls the step size during the Q-value updates and how much the neural network weights are updated during training, while γ determines the importance of future rewards, where values close to 1 give more weight to future rewards, whereas values close to 0 focus on immediate rewards. ε determines the probability of choosing a random action for the exploration–exploitation strategy; over time, ε is reduced to favor exploitation using ε-decay. For the Experience Replay, its size is crucial, as it should be large enough to store a representative set of experiences. The Minibatch Size should be chosen properly when sampling from the Experience Replay. It specifies the number of experiences in each Minibatch. The frequency of updating the Target Network is every 10,000 steps to avoid instability, and the mean squared error (MSE) is used as the loss function between the predicted and target Q-values.

These hyperparameters were experimented, fine-tuned, and adapted after gaining insights from training and evaluation to achieve the best performance in terms of balancing between exploration and exploitation, which, in turn, optimizes the performance of learning and the convergence of the Deep Q-Learning algorithm.

We have configured the neural network here with four layers, where the two hidden layers have 512 and 64 neurons, using the Rectified Linear Unit activation function and the Epsilon-Greedy action selection algorithm to choose between exploration and exploitation randomly [50]. We used a workload dataset, Azur workload 2019, to train the model [51]. It contains the following information: min, max, and avg CPU utilization, VM id, VM

category, VM core count, VM memory, count VMs created, deployment size, and Timestamp VM created and deleted. We have manipulated this dataset to merge the 1200 resources we collected using the *RC* module, as per the categorization mentioned above, along with the assumed reputation values for the experiment purpose and to prove the concept. Therefore, the Predictor learned how to differentiate between the available candidates to select the most appropriate one to be the executor for the job. The job here is assumed to be any of the workloads from the dataset.

Table 2. Experimental parameter settings.

Notation	Parameter	Configuration
-	Number of episodes	1000 episodes
α	Learning rate	0.01
γ	Discount factor	0.95
ε	Epsilon-Greedy parameter	1.0
ε-decay	Reducing factor of exploration rate	0.995
ER	Experience Replay	10,000 steps
-	Minibatch Size	32 experiences
TRX	Native token of Tron Blockchain	Tron Blockchain
Tron	Blockchain network	Testnet network
SR Node	Blockchain super representative node	1 node
Candidate	Blockchain participant nodes	7 candidates
Q	Reward of successful execution per task	1 TRX
ψ^I	Size of each task	500,000 bytes
x	Computation intensity	5 cycles/byte
wS	Weight of stake	0.05
V	Reputation value	0.1

The experiment was conducted using a system with an Intel Core i7 CPU @ 2.30 GHz processor and 16 GB RAM, and we used several technical tools and frameworks including: Python (3.7), Tronpy (0.4.0), Azure-core (1.29.5), Google-api-core (2.12.0), AWS Boto3 (1.28.61), Gymnasium (0.29.1), Numpy (1.26.0), Pandas (2.1.1), OpenCV-python (4.8.1.78), Mpmath (1.3.0), Pylint (3.0.1), PyCryptodome (3.19.0), Psutil (5.9.6), Pyflakes (3.1.0), cryptography (41.0.4), Matplotlib (3.8.0), PyTorch (2.1.0), and others.

4.2. Performance Evaluation

Figure 6 shows the performance of the proposed *Predictor*, where the total reward is noticed to be low at the beginning of learning then becomes higher as the number of decision epochs increases. We can notice that after around 400 decision epochs, it becomes stable. This fluctuation is a result of the changes in the environment, which is the state space, affecting the obtained reward in different decision epochs.

We validated the proposed mechanism for selecting suitable executors θ_E from the candidate set θ_C. We took a snapshot of θ_E from the available θ_C at one decision epoch. Figures 7–9 show how the proposed algorithm has selected only five suitable executors from the seven candidates available, based on the workload nature, the executors' stake wS, and the reputation value V. Hence, the executor selection action is $a = \{1, 0, 1, 1, 1, 0, 1\}$. This ensures a trustworthy evaluation of the potential candidates before selecting them.

At the same decision epoch, candidate 5 is assigned with 20 tasks (chunks of the job workload), as shown in Figure 8, and this candidate utilizes 2.8 GHz of computing power to completely execute the job, as shown in Figure 9. The same goes for candidate 7, where only 10 tasks were assigned to this candidate, and it utilizes 1.7 GHz of computing power. This proves that while the Predictor aims to achieve the highest total execution reward possible, it also carefully evaluates the candidates and schedules tasks accordingly.

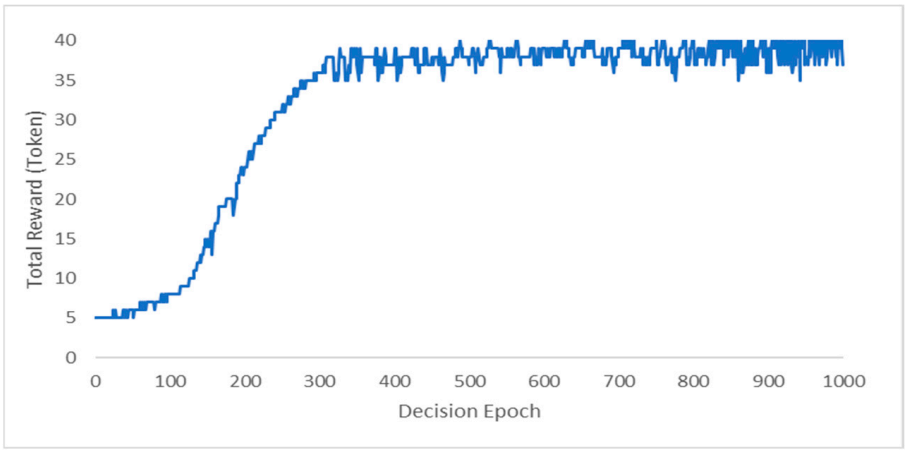

Figure 6. Performance of the proposed Predictor.

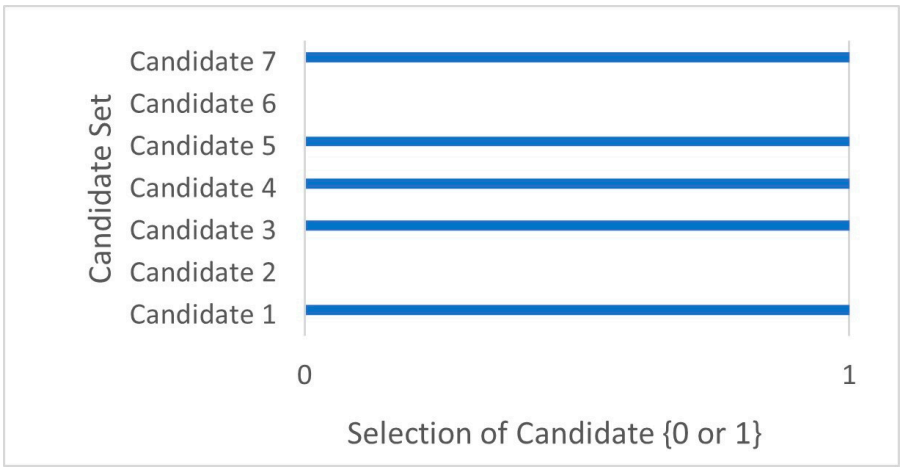

Figure 7. Selected candidates.

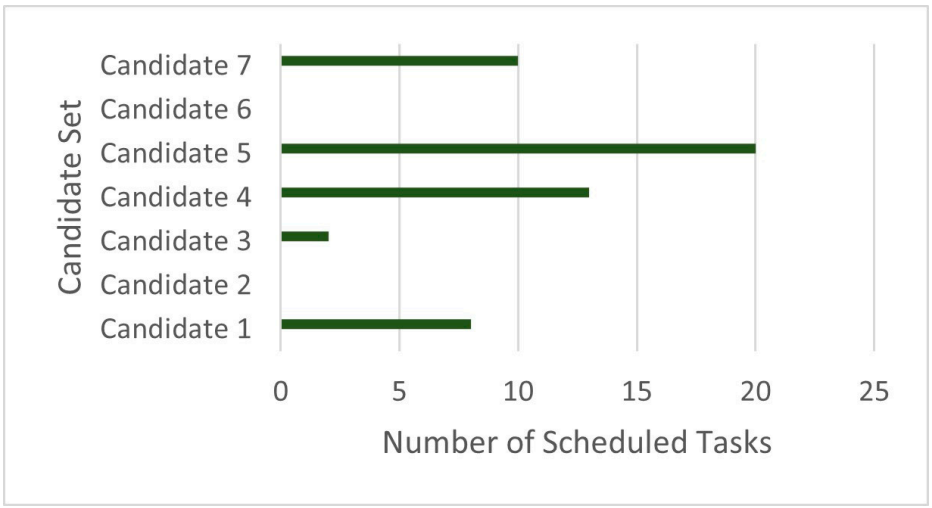

Figure 8. Scheduled tasks per candidate.

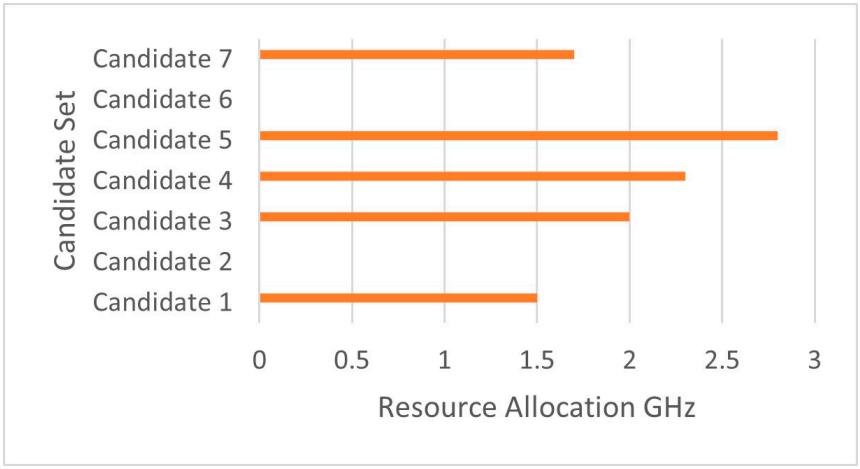

Figure 9. Allocated resources per candidate.

We can also interpret from this snapshot that because of the different computing capabilities of the different executors, the resource allocation policy is not uniform. Table 3 summarizes the snapshot and shows how executors are selected based on the assessment of individual candidates' capabilities, showing how dynamic and selective the Predictor is and that its policies are not uniform.

Table 3. A snapshot of θ_E from the available θ_C at one decision epoch.

Candidate	Selection of Candidate	Number of Scheduled Tasks	Resource Allocation
1	1	8	1.5
2	0	0	0
3	1	2	2
4	1	13	2.3
5	1	20	2.8
6	0	0	0
7	1	10	1.7

To test the computational resource allocation capability of the Predictor and the effect of the QoS requirements, we examined the performance in terms of the total consumed computational resources. Figure 10 depicts how the Predictor managed to save the consumed computational resources with varying timeout $\mathcal{T}$ constraints. Evidently, fewer computing resources are usually required when we increase the timeout and have a flexible QoS.

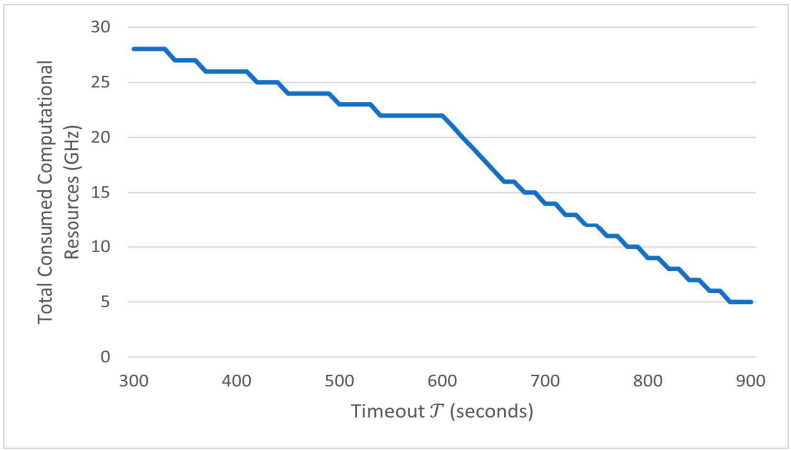

Figure 10. Total consumed computational resources as timeout constraint varies.

To validate the proposed idea in a real-world federated cloud environment, we assumed the required job is related to "dataset processing", which is resource-intensive and requires high memory. The dataset used in this test case was the *Titanic Dataset* [52], as a CSV file contains all passengers' data with the purpose of processing and analyzing by pulling out different fields and calculating the average of different subfields, so multiple mathematical operations were required on the data. We evaluated the performance here by just completing the job successfully on a federated cloud that included AWS, GCP, and Azure cloud providers, as this proves that the Predictor has chosen suitable executors (here, VMs) for the job by considering their computing capabilities and job type, ensuring a successful job execution to minimize the possibility of increased execution time that might violate the assumed QoS requirement, here $\mathcal{T} \leq 600$ s, hence avoiding aborting the job before completion. So, here we initialized 15 VMs as candidates θ_C, with hypothetical wS and V values for each, 5 VMs from each cloud provider, and all being participants in Tron Blockchain with the help of a tool called Kaleido [53] to securely manage and share files in Blockchain. We measured the total completion time for the job and compared it with a flexible case as a baseline where there is no constraint on the timeout.

This indicates that the proposed DQL-based Predictor has optimized its selection of θ_E from the 15 candidates available in θ_C and met the required QoS, $\mathcal{T} \leq 600$ s. As shown in Figure 11, the time taken without considering the QoS is 648 s, and only 553 s using the proposed mechanism that considers the QoS to complete the same job.

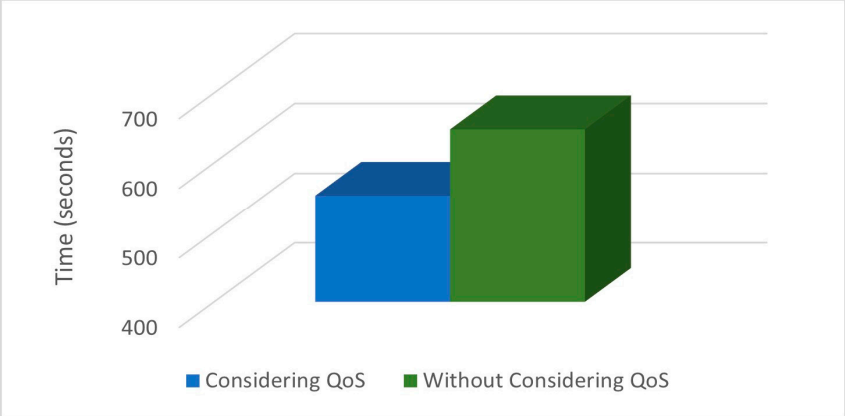

Figure 11. Calculation of the total job completion time.

As the proposed system should theoretically allow for services from third parties like independent individual service providers, we can ensure that it is not possible for a malicious service provider (candidate executor) to join the network by adopting the Delegated Proof-of-Stake (DPoS) as the consensus mechanism in the Blockchain. Hence, the executor trust is guaranteed here through the strict verification procedure of the DPoS, which ensures that the elected nodes responsible for signing blocks are doing that in an unbiased way. So, unlike other consensus mechanisms, there is no need to depend on untrusted nodes to verify a transaction before confirming that transaction using the DPoS [54]. Also, every signed block will have to verify that it is generated from a trusted node. As per the voting system in DPoS, all voting is carried out in real-time, so any malicious action can be immediately detected. In the implementation of this research, we used TRON as it is a DPoS network, where Super Representatives (SRs) are responsible for validating transactions and record keeping, and in case of inefficient or unavailable SRs, voters can switch over their votes to a better node to guarantee network security.

4.3. Discussion

From the above results and evaluation, the proposed DQL-based mechanism in this system has shown a high level of accuracy in predicting the proper task scheduling in

the federated cloud where the state space is a decentralized environment; it handled this dynamicity and achieved a high reward for executing jobs while satisfying the QoS requirements. Generally, the performance of hyper-heuristics depends on the nature of the problem and the current state of the solution, meaning that in addition to the solution quality and feedback mechanisms, the variability in problems (domain and complexity) highlight the adaptability and flexibility of hyper-heuristics to cater to a wide array of problems.

When we tested the Predictor without *BCM* (ignoring wS and V values), we found that sometimes, a set of candidates were selected as executors for specific job workloads, but also at other times, they were not selected for the same workloads under the same conditions and QoS requirements. This is because the candidates' cumulative stakes and reputation values were not evaluated before selection, and this confirms how the weight of stake and candidate reputation affects the selection criteria while the Predictor is selecting the *executors*. On the other hand, when we assumed a higher weight for wS, ranging from 0.05 to 0.9, the reward increased, as implied by Equation (8), which aims to maximize the reward for executors while satisfying the QoS requirements.

This proof-of-concept shows how Blockchain here enabled our proposed DQL-based system to be considered as a token-based incentive mechanism for executors to offer their various services, hence allowing us to decentralize the processing of jobs on federated clouds in a transparent way that is based on automated smart contracts, avoiding any possibility of a biased selection of candidates that is not based on their reputation or stakes.

It is worth mentioning that transaction cost in Tron is very high. We noticed that every function call to the Blockchain and smart contracts regarding any job transactions consumes too much energy and bandwidth, causing it to run out of energy and bandwidth so fast that it needs to stake more assets (*TRX*) for resources and Tron power to obtain more energy and bandwidth. In one case, a smart contract was called only three times and consumed the full bandwidth available, 1500B Bandwidth Points (BP), $1500/1024$ KB $\approx$ 1.46 KB, and we had to stake (freeze) at least 3000 TRX to obtain more bandwidth, with a 3-day period for un-staking to redeem the frozen TRX. Even though the way the smart contract is built is very memory optimized, the transaction cost is still very high. However, this issue can be resolved, or at least minimized, by considering different platforms other than Tron that have an acceptable transaction cost.

Other limitations encountered during the research include the significant computational resources required for training and the difficult integration of Blockchain and DQL, which requires careful consideration of interoperability between different components. Moreover, the complexity of task scheduling, as an NP-hard problem, is compounded here in this tokenized system, as the Blockchain operations add another layer of operational complexity. Although it needs a lot of configurations, integrations, and technical tools to get this solution working in a basic form as a proof-of-concept, we can say it is promising, and it opens the possibilities of different advanced use cases, considering that further fine-tuning is required to get a better performance.

5. Conclusions

The proposed system of employing Deep Q-Learning with Blockchain for tokenized task scheduling aims to combine the power of DQL with the transparent and decentralized nature of Blockchain to enable a tokenized system that can solve, to some extent, the vendor lock-in issue that is related to cloud providers and the potential security considerations related to heterogeneous services from third parties like individuals or other cloud providers that offer their services based on tokens. From the discussed results of this proposed proof-of-concept, we can conclude that it is capable of offering:

- Efficient task scheduling and resource allocation by leveraging the DQL for more optimization; such algorithms excel at learning from interactions and making sequential decisions based on the requirements of different tasks through adapting to the dynamic conditions, resulting in an optimized performance and utilization;

- Tokenized incentives and rewards for independent service providers (*executors*) who contribute their resources or compute power to task execution to ensure fairness;
- Transparent and trustworthy task execution through the utilization of the immutable Blockchain technology, which helps with verifying the execution of tasks, enhancing trust and reducing the reliance on centralized authorities and single cloud providers;
- Distributed and secured task execution responsibilities among participants by leveraging Blockchain's decentralized nature. This enhances system resilience, reduces single points of failure, and improves security against malicious activities.

For future work, we will try to optimize the system for more specialized fields, such as digital forensics, Decentralized Finance (DeFi), and healthcare systems. In addition, we will try to find other Blockchain platforms with less expensive transactions that can support the future demand for Blockchain enabled AI systems.

Author Contributions: Conceptualization, O.Y. and F.E.; methodology, O.Y., F.E. and L.E.; validation, K.J. and F.E.; formal analysis, O.Y., K.J., F.E. and L.E.; investigation, O.Y.; resources, K.J., F.E. and L.E.; data curation, O.Y.; writing—original draft preparation, O.Y.; writing—review and editing, O.Y. and F.E.; supervision, K.J.; project administration, K.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [github.com/Azure/AzurePublicDataset/blob/master/AzurePublicDatasetV2.md], (accessed on 2 March 2024).

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

References

1. Barreto, L.; Fraga, J.; Siqueira, F. Conceptual Model of Brokering and Authentication in Cloud Federations. In Proceedings of the 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), Niagara Falls, ON, Canada, 5–7 October 2015; pp. 303–308.
2. Bohn, R.B.; Chaparadza, R.; Elkotob, M.; Choi, T. The Path to Cloud Federation through Standardization. In Proceedings of the 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 942–946.
3. Zhao, Y.; Zhang, J.; Cao, Y. Manipulating Vulnerability: Poisoning Attacks and Countermeasures in Federated Cloud–Edge–Client Learning for Image Classification. *Knowl.-Based Syst.* **2023**, *259*, 110072. [CrossRef]
4. Verma, A.; Bhattacharya, P.; Bodkhe, U.; Saraswat, D.; Tanwar, S.; Dev, K. FedRec: Trusted Rank-Based Recommender Scheme for Service Provisioning in Federated Cloud Environment. *Digit. Commun. Netw.* **2023**, *9*, 33–46. [CrossRef]
5. Magdy, Y.; Azab, M.; Hamada, A.; Rizk, M.R.M.; Sadek, N. Moving-Target Defense in Depth: Pervasive Self- and Situation-Aware VM Mobilization across Federated Clouds in Presence of Active Attacks. *Sensors* **2022**, *22*, 9548. [CrossRef] [PubMed]
6. Badshah, A.; Ghani, A.; Siddiqui, I.F.; Daud, A.; Zubair, M.; Mehmood, Z. Orchestrating Model to Improve Utilization of IaaS Environment for Sustainable Revenue. *Sustain. Energy Technol. Assess.* **2023**, *57*, 103228. [CrossRef]
7. Alharbe, N.; Aljohani, A.; Rakrouki, M.A.; Khayyat, M. An Access Control Model Based on System Security Risk for Dynamic Sensitive Data Storage in the Cloud. *Appl. Sci.* **2023**, *13*, 3187. [CrossRef]
8. Alashhab, Z.R.; Anbar, M.; Singh, M.M.; Hasbullah, I.H.; Jain, P.; Al-Amiedy, T.A. Distributed Denial of Service Attacks against Cloud Computing Environment: Survey, Issues, Challenges and Coherent Taxonomy. *Appl. Sci.* **2022**, *12*, 12441. [CrossRef]
9. Kollu, V.N.; Janarthanan, V.; Karupusamy, M.; Ramachandran, M. Cloud-Based Smart Contract Analysis in FinTech Using IoT-Integrated Federated Learning in Intrusion Detection. *Data* **2023**, *8*, 83. [CrossRef]
10. Pol, P.S.; Pachghare, V.K. A Review on Trust-Based Resource Allocation in Cloud Environment: Issues Toward Collaborative Cloud. *Int. J. Semant. Comput.* **2023**, *17*, 59–91. [CrossRef]
11. Lee, J.; Kim, B.; Lee, A.R. Priority Evaluation Factors for Blockchain Application Services in Public Sectors. *PLoS ONE* **2023**, *18*, e0279445. [CrossRef]
12. Krichen, M.; Ammi, M.; Mihoub, A.; Almutiq, M. Blockchain for Modern Applications: A Survey. *Sensors* **2022**, *22*, 5274. [CrossRef]
13. Tyagi, A.K.; Dananjayan, S.; Agarwal, D.; Thariq Ahmed, H.F. Blockchain—Internet of Things Applications: Opportunities and Challenges for Industry 4.0 and Society 5.0. *Sensors* **2023**, *23*, 947. [CrossRef] [PubMed]
14. Nawrocki, P.; Osypanka, P.; Posluszny, B. Data-Driven Adaptive Prediction of Cloud Resource Usage. *J. Grid Comput.* **2023**, *21*, 6. [CrossRef]

15. Abdel-Hamid, L. An Efficient Machine Learning-Based Emotional Valence Recognition Approach Towards Wearable EEG. *Sensors* **2023**, *23*, 1255. [CrossRef] [PubMed]
16. Lei, B.; Zhou, J.; Ma, M.; Niu, X. DQN Based Blockchain Data Storage in Resource-Constrained IoT System. In Proceedings of the 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, UK, 26–29 March 2023; pp. 1–6. [CrossRef]
17. Katib, I.; Assiri, F.Y.; Althaqafi, T.; AlKubaisy, Z.M.; Hamed, D.; Ragab, M. Hybrid Hunter–Prey Optimization with Deep Learning-Based Fintech for Predicting Financial Crises in the Economy and Society. *Electronics* **2023**, *12*, 3429. [CrossRef]
18. Albeshri, A. SVSL: A Human Activity Recognition Method Using Soft-Voting and Self-Learning. *Algorithms* **2021**, *14*, 245. [CrossRef]
19. Al-Ghamdi, A.S.A.-M.; Ragab, M.; AlGhamdi, S.A.; Asseri, A.H.; Mansour, R.F.; Koundal, D. Detection of Dental Diseases through X-Ray Images Using Neural Search Architecture Network. *Comput. Intell. Neurosci.* **2022**, *2022*, 3500552. [CrossRef] [PubMed]
20. Jambi, K.M.; Khan, I.H.; Siddiqui, M.A. Evaluation of Different Plagiarism Detection Methods: A Fuzzy MCDM Perspective. *Appl. Sci.* **2022**, *12*, 4580. [CrossRef]
21. Denizdurduran, B.; Markram, H.; Gewaltig, M.-O. Optimum Trajectory Learning in Musculoskeletal Systems with Model Predictive Control and Deep Reinforcement Learning. *Biol. Cybern.* **2022**, *116*, 711–726. [CrossRef]
22. Majid, A.Y.; Saaybi, S.; Francois-Lavet, V.; Prasad, R.V.; Verhoeven, C. Deep Reinforcement Learning Versus Evolution Strategies: A Comparative Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–19. [CrossRef]
23. Tefera, M.K.; Zhang, S.; Jin, Z. Deep Reinforcement Learning-Assisted Optimization for Resource Allocation in Downlink OFDMA Cooperative Systems. *Entropy* **2023**, *25*, 413. [CrossRef]
24. Özcan, E.; Drake, J.H.; Burke, E.K. A Modified Choice Function Hyper-Heuristic Controlling Unary and Binary Operators. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015.
25. Ortiz-Bayliss, J.C.; Terashima-Marín, H.; Conant-Pablos, S.E. Combine and Conquer: An Evolutionary Hyper-Heuristic Approach for Solving Constraint Satisfaction Problems. *Artif. Intell. Rev. Int. Sci. Eng. J.* **2016**, *46*, 327–349. [CrossRef]
26. Espinoza-Nevárez, D.; Ortiz-Bayliss, J.C.; Terashima-Marín, H.; Gatica, G. Selection and Generation Hyper-Heuristics for Solving the Vehicle Routing Problem with Time Windows. In Proceedings of the GECCO 2016 Companion—Genetic and Evolutionary Computation Conference, Denver, CO, USA, 20–24 July 2016; Association for Computing Machinery, Inc.: New York, NY, USA, 2016; pp. 139–140.
27. Zubaydi, H.D.; Varga, P.; Molnár, S. Leveraging Blockchain Technology for Ensuring Security and Privacy Aspects in Internet of Things: A Systematic Literature Review. *Sensors* **2023**, *23*, 788. [CrossRef] [PubMed]
28. Alam, T. Blockchain-Based Internet of Things: Review, Current Trends, Applications, and Future Challenges. *Computers* **2023**, *12*, 6. [CrossRef]
29. Samy, A.; Elgendy, I.A.; Yu, H.; Zhang, W.; Zhang, H. Secure Task Offloading in Blockchain-Enabled Mobile Edge Computing with Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 4872–4887. [CrossRef]
30. Lin, K.; Gao, J.; Han, G.; Wang, H.; Li, C. Intelligent Blockchain-Enabled Adaptive Collaborative Resource Scheduling in Large-Scale Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2022**, *18*, 9196–9205. [CrossRef]
31. Xiao, H.; Qiu, C.; Yang, Q.; Huang, H.; Wang, J.; Su, C. Deep Reinforcement Learning for Optimal Resource Allocation in Blockchain-Based IoV Secure Systems. In Proceedings of the 2020 16th International Conference on Mobility, Sensing and Networking (MSN), Tokyo, Japan, 17–19 December 2020; pp. 137–144.
32. Gao, S.; Wang, Y.; Feng, N.; Wei, Z.; Zhao, J. Deep Reinforcement Learning-Based Video Offloading and Resource Allocation in NOMA-Enabled Networks. *Future Internet* **2023**, *15*, 184. [CrossRef]
33. Fang, C.; Zhang, T.; Huang, J.; Xu, H.; Hu, Z.; Yang, Y.; Wang, Z.; Zhou, Z.; Luo, X. A DRL-Driven Intelligent Optimization Strategy for Resource Allocation in Cloud-Edge-End Cooperation Environments. *Symmetry* **2022**, *14*, 2120. [CrossRef]
34. Quan, T.; Zhang, H.; Yu, Y.; Tang, Y.; Liu, F.; Hao, H. Seismic Data Query Algorithm Based on Edge Computing. *Electronics* **2023**, *12*, 2728. [CrossRef]
35. Todorović, M.; Matijević, L.; Ramljak, D.; Davidović, T.; Urošević, D.; Jakšić Krüger, T.; Jovanović, Đ. Proof-of-Useful-Work: BlockChain Mining by Solving Real-Life Optimization Problems. *Symmetry* **2022**, *14*, 1831. [CrossRef]
36. Cheng, Y.; Cao, Z.; Zhang, X.; Cao, Q.; Zhang, D. Multi Objective Dynamic Task Scheduling Optimization Algorithm Based on Deep Reinforcement Learning. *J. Supercomput. Int. J. High-Perform. Comput. Des. Anal. Use* **2023**, *79*, 1–29. [CrossRef]
37. Jain, V.; Kumar, B. QoS-Aware Task Offloading in Fog Environment Using Multi-Agent Deep Reinforcement Learning. *J. Netw. Syst. Manag.* **2022**, *31*, 7. [CrossRef]
38. Liu, H.; Zhou, H.; Chen, H.; Yan, Y.; Huang, J.; Xiong, A.; Yang, S.; Chen, J.; Guo, S. A Federated Learning Multi-Task Scheduling Mechanism Based on Trusted Computing Sandbox. *Sensors* **2023**, *23*, 2093. [CrossRef] [PubMed]
39. Lakhan, A.; Mohammed, M.A.; Nedoma, J.; Martinek, R.; Tiwari, P.; Kumar, N. DRLBTS: Deep Reinforcement Learning-Aware Blockchain-Based Healthcare System. *Sci. Rep.* **2023**, *13*, 4124. [CrossRef] [PubMed]
40. Dong, Y.; Alwakeel, A.M.; Alwakeel, M.M.; Alharbi, L.A.; Althubiti, S.A. A Heuristic Deep Q Learning for Offloading in Edge Devices in 5 g Networks. *J. Grid Comput.* **2023**, *21*, 37. [CrossRef]
41. Neves, M.; Vieira, M.; Neto, P. A Study on a Q-Learning Algorithm Application to a Manufacturing Assembly Problem. *J. Manuf. Syst.* **2021**, *59*, 426–440. [CrossRef]
42. Gao, J.; Niu, K. A Reinforcement Learning Based Decoding Method of Short Polar Codes. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March 2021; pp. 1–6.

43. Kardani-Moghaddam, S.; Buyya, R.; Ramamohanarao, K. ADRL: A Hybrid Anomaly-Aware Deep Reinforcement Learning-Based Resource Scaling in Clouds. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 514–526. [CrossRef]
44. Yi, D.; Zhou, X.; Wen, Y.; Tan, R. Efficient Compute-Intensive Job Allocation in Data Centers via Deep Reinforcement Learning. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1474–1485. [CrossRef]
45. Zhang, S.; Lee, J.-H. Analysis of the Main Consensus Protocols of Blockchain. *ICT Express* **2020**, *6*, 93–97. [CrossRef]
46. Liu, G.; Chen, C.-Y.; Han, J.-Y.; Zhou, Y.; He, G.-B. NetDAO: Toward Trustful and Secure IoT Networks without Central Gateways. *Symmetry* **2022**, *14*, 1796. [CrossRef]
47. Ahamed, Z.; Khemakhem, M.; Eassa, F.; Alsolami, F.; Basuhail, A.; Jambi, K. Deep Reinforcement Learning for Workload Prediction in Federated Cloud Environments. *Sensors* **2023**, *23*, 6911. [CrossRef]
48. TRON | Decentralize the Web. Available online: https://tron.network/ (accessed on 18 July 2023).
49. Delegated Proof of Stake (DPOS)—BitShares Documentation. Available online: https://how.bitshares.works/en/master/technology/dpos.html (accessed on 20 July 2023).
50. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *Neurocomputing* **2022**, *503*, 92–108. [CrossRef]
51. AzurePublicDatasetV2. Available online: https://github.com/Azure/AzurePublicDataset/blob/master/AzurePublicDatasetV2.md (accessed on 4 August 2023).
52. Datasets/Titanic.Csv at Master Datasciencedojo/Datasets. Available online: https://github.com/datasciencedojo/datasets/blob/master/titanic.csv (accessed on 1 October 2023).
53. Blockchain File Sharing & Storage—Kaleido Document Exchange. Available online: https://www.kaleido.io/blockchain-platform/document-exchange (accessed on 16 October 2023).
54. Bachani, V.; Bhattacharjya, A. Preferential Delegated Proof of Stake (PDPoS)—Modified DPoS with Two Layers towards Scalability and Higher TPS. *Symmetry* **2023**, *15*, 4. [CrossRef]