IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 21 December 2022, accepted 28 December 2022, date of publication 30 December 2022, date of current version 5 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3233474

RESEARCH ARTICLE

Adaptive Storage Optimization Scheme for Blockchain-IIoT Applications Using Deep Reinforcement Learning

NANA KWADWO AKRASI-MENSAH[1,2], ANDREW SELASI AGBEMENU[1,2,3], (Member, IEEE), HENRY NUNOO-MENSAH[1,2,3], (Member, IEEE), ERIC TUTU TCHAO[1,2,3], (Member, IEEE), ABDUL-RAHMAN AHMED[1,4], (Member, IEEE), ELIEL KEELSON[1,2], AXEL SIKORA[5], DOMINIK WELTE[5], AND JERRY JOHN KPONYO[3,4], (Member, IEEE)

[1]Distributed IoT Platforms, Privacy and Edge-Intelligence Research (DIPPER) Laboratory, Faculty of Electrical and Computer Engineering, Kwame Nkrumah University of Science and Technology, PMB, Kumasi, Ghana
[2]Department of Computer Engineering, Kwame Nkrumah University of Science and Technology, PMB, Kumasi, Ghana
[3]Responsible Artificial Intelligence Laboratory (RAIL), Faculty of Electrical and Computer Engineering, Kwame Nkrumah University of Science and Technology, PMB, Kumasi, Ghana
[4]Department of Telecommunications Engineering, Kwame Nkrumah University of Science and Technology, PMB, Kumasi, Ghana
[5]Institute of Reliable Embedded Systems and Communication Electronics (ivESK), Offenburg University of Applied Sciences, 77652 Offenburg, Germany

Corresponding authors: Andrew Selasi Agbemenu (asagbemenu@knust.edu.gh) and Nana Kwadwo Akrasi-Mensah (nakrasi97@gmail.com)

This work was supported in part by the German Federal Ministry of Research and Education (Bundesministerium für Bildung und Forschung, BMBF) and the German Academic Exchange Service (Deutscher Akademischer Austauschdienst, DAAD); in part by the Distributed Internet of Things (IoT)-Platforms for Safe Food Production in Education, Research and Industry (DIPPER) Project, which is co-financed by the BMBF (Förderkennzeichen: 01DG21017) and DAAD under Grant 57557211; and in part by the Responsible Artificial Intelligence Laboratory Project which is being co-financed by the International Development Research Centre (IDRC) under Grant 109832-001 and Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) under Agreement 81280702.

ABSTRACT Blockchain-IIoT integration into industrial processes promises greater security, transparency, and traceability. However, this advancement faces significant storage and scalability issues with existing blockchain technologies. Each peer in the blockchain network maintains a full copy of the ledger which is updated through consensus. This full replication approach places a burden on the storage space of the peers and would quickly outstrip the storage capacity of resource-constrained IIoT devices. Various solutions utilizing compression, summarization or different storage schemes have been proposed in literature. The use of cloud resources for blockchain storage has been extensively studied in recent years. Nonetheless, block selection remains a substantial challenge associated with cloud resources and blockchain integration. This paper proposes a deep reinforcement learning (DRL) approach as an alternative to solving the block selection problem, which involves identifying the blocks to be transferred to the cloud. We propose a DRL approach to solve our problem by converting the multi-objective optimization of block selection into a Markov decision process (MDP). We design a simulated blockchain environment for training and testing our proposed DRL approach. We utilize two DRL algorithms, Advantage Actor-Critic (A2C), and Proximal Policy Optimization (PPO) to solve the block selection problem and analyze their performance gains. PPO and A2C achieve 47.8% and 42.9% storage reduction on the blockchain peer compared to the full replication approach of conventional blockchain systems. The slowest DRL algorithm, A2C, achieves a run-time 7.2 times shorter than the benchmark evolutionary algorithms used in earlier works, which validates the gains introduced by the DRL algorithms. The simulation results further show that our DRL algorithms provide an adaptive and dynamic solution to the time-sensitive blockchain-IIoT environment.

INDEX TERMS Blockchain, IIoT, reinforcement learning, scalability, storage efficiency, storage optimization.

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

I. INTRODUCTION

The enormous popularity of cryptocurrencies like Bitcoin [1] and Ethereum [2] enabled the development of blockchain technology as a secure solution for the Industrial Internet of Things (IIoT). There are applications for blockchain in a variety of industries, including health [3], [4], [5], agriculture [6], [7], supply chain management [8], education [9], real estate [10], [11] and manufacturing [12]. The blockchain's immutability, consensus, and smart contract capabilities impose a robust architecture that does away with third-party verification requirements, enables auditing and traceability, and promotes transactions between various parties. In the blockchain, a network of peers maintain a decentralized ledger as each peer stores a full copy of the ledger. The ledger can only be appended to and is updated through the consensus of the peers in the network. A party performing a transaction on the blockchain does not need to establish trust with the other party through a third party since the blockchain's cryptographic nature and consensus ensures trust. Industrial processes that demand cooperation between many entities, like supply chains, may benefit from these capabilities [8]. Over time, data accumulates on the blockchain due to its immutable nature. The blockchain network's performance and scalability are affected by the unchecked growth of the blockchain ledger. Data generation may exceed the peers' capacity to store blocks in the high transaction environment of IIoT networks, which would limit the involvement of devices with limited resources, such as low-power IoT devices. The expansion of the blockchain ledger impacts the time it takes a new node to integrate into the network entirely and impacts the efficiency of the underpinning databases, causing delays during database reading and writing.

Literature has proposed utilizing cloud storage systems to mitigate the storage pressure on peers in IoT systems. However, this increases latency and impacts the blockchain's performance [13]. The authors of [13] devised the non-dominated sorting genetic algorithm with clustering (NSGA-C) to discover the ideal number of blocks to move to the cloud. Nartey et al. [14] addressed the problem of block selection by presenting the advanced time-variant multi-objective particle swarm optimization (AT-MOPSO) algorithm. The algorithms proposed in [13] and [14] belong to a class of nature-inspired meta-heuristic algorithms known as multi-objective evolutionary algorithms (MOEAs), which are commonly employed to solve complex multi-objective optimization problems. Issues such as lengthy run-times and a lack of sensitivity to variations hinder the performance of these algorithms [15].

This paper provides a deep reinforcement learning (DRL) solution to the block selection problem. We propose an adaptive approach, in which a DRL agent on a peer moves blocks between the cloud and local storage depending on the organization's needs and the network state to ensure optimal blockchain performance. In our method, a reward function for our DRL agent integrates the three objective functions of the query probability of blocks in cloud storage, the cost of cloud storage, and the local space occupancy or storage availability of peers. The minimization of these objective functions maximizes rewards, encouraging smarter decisions about which blocks to transfer to the cloud. The following is a summary of the main contributions of this paper:

1) An adaptive optimization scheme is proposed to reduce the storage demand on blockchain peers while optimizing query cost in blockchain-IIoT applications using DRL. In our approach, bi-directional block movement allows a DRL agent to adapt to the constantly changing query frequency of blocks in specific applications.
2) The block selection problem is defined as a Markov Decision Process (MDP). We model the blockchain environment in terms of the states of the blocks on the blockchain, the actions available to the agent and the rewards that actions taken result in.
3) The proposed approach is evaluated using state-of-the-art DRL algorithms. DRL models are trained on a simulated environment to solve the block selection problem for cloud storage and the performance of the models are evaluated.

The remainder of this paper is structured as follows: The related works are presented in Section II, the system model is presented in Section III, the problem formulation and proposed approach are discussed in Section IV, the evaluation of our experimental findings is presented in Section V and the study is concluded in Section VI.

II. RELATED WORKS

Numerous strategies to optimize storage in blockchain systems have been proposed in the literature [16]. Some of these methods include compression and summarization techniques to minimize the number of blocks on the blockchain, hence decreasing the total size of the ledger [17], [18], [19], [20], [21]. Other methods suggest alternatives to the typical blockchain's full replication storage approach. One of these methods is to distribute blocks among peers in the network based on their relevance to the peer's operation and resources so that other peers can request blocks they do not keep locally. In the sharding-based strategy, various shards of peers within the blockchain each maintain their ledger and only process transactions of their shard [22]. Some works partition the blockchain network into storage units, in which component peers donate storage resources to store the entire ledger and query within their storage unit for blocks they do not store locally [23], [24], [25]. Another intuitive concept discussed in depth in this study is optimizing blockchain storage via cloud storage [13], [14]. Several works have recommended evolutionary algorithms to supplement this approach to ensure that cloud-based block storage maintains optimal blockchain performance.

A. MULTI-OBJECTIVE OPTIMIZATION IN BLOCKCHAIN SYSTEMS

Multi-objective optimization challenges are very ubiquitous in the real world. They require effective decision-making in the face of several conflicting objectives. Evolutionary algorithms and handcrafted heuristics have been widely applied as iteration-based problem solvers to solve multi-objective optimization problems. Evolutionary algorithms' population-based nature enables them to generate multiple solutions in a single run efficiently. However, the performance of these classical methods is hindered by constraints that have been extensively reviewed in the literature [15], [26]. First, for problems with large dimensions, the number of iterations required for population update or iterative searching to obtain near-optimal solutions sometimes leads to long execution times. Another difficulty with these strategies is their failure to account for problem variability. Since they are frequently tailored to the problem to which they are applied, even minor modifications necessitate a reapplication of the approach to produce satisfactory results. This trait is elaborated upon in the No Free Lunch Theorem [27]. These techniques are impractical for usage in applications with stringent latency requirements.

Xu et al. [13] first described the storage optimization of the blockchain for IoT systems as a block selection problem for cloud storage. They proposed the NSGA-C algorithm to perform this task. They reduced storage overhead by an average of 30%. However, their approach had a comparatively high run-time compared to the benchmarks. To improve the work of [13], Nartey et al. [14] introduced the AT-MOPSO algorithm. The AT-MOPSO algorithm had a significantly shorter run-time and greater energy efficiency than the NSGA-C. However, this algorithm fared worse on one of the trade-off objectives than NSGA-C.

B. DEEP REINFORCEMENT LEARNING

Even though classical optimization techniques, such as evolutionary algorithms and handcrafted heuristics, are suitable for enhancing performance, recent advances in machine learning have led to the development of algorithms that look promising for solving a wide range of computational problems. We propose DRL as an alternative to the evolutionary algorithms introduced in [13] and [14] for solving the multi-objective optimization problem of block selection for storage optimization in blockchain-IIoT systems. This approach may offer benefits over evolutionary algorithms and custom heuristics such as [15]:

- *Shorter run-time*: Solutions can be directly obtained by a simple forward propagation of the model once a trained model is available.
- *Strong generalization ability*: The nature of the DRL ensures it can be used for new instances of the problem without modifications.
- *Scalability*: As demonstrated by Li et al. [15], the DRL approach may scale better for a large number of blockchain peers.

RL focuses on sequential decision-making in dynamic, partly controlled environments. A reinforcement learning problem consists of two fundamental elements: an agent and an environment. The agent's interactions with the environment are geared towards maximizing an objective, defined as the cumulative reward obtained by the agent through its actions. The agent collects observations and rewards from the environment and utilizes this data to build a policy, which is a function that maps the states of the environment to actions. The agent's action influences the environment, transitioning to a new state. Repeated interactions with the environment offer the essential data for the reinforcement learning algorithm to discover the best policy that maximizes the cumulative reward.

DRL blends deep learning and reinforcement learning. Reinforcement learning is concerned with agents that learn via interactions with their environment, without understanding of the system model, in order to maximize long-term performance or accomplish a specific goal. Deep learning has its roots in early attempts to mimic the networks of neurons in the brain using computational circuits [28]. It involves the use of artificial neural networks that are often multilayered for high-dimensional sensory representation and are well-suited for universal approximation. DRL employs deep learning as a method for function approximation to escape the curse of dimensionality [29].

Recently, DRL has been discussed in literature as an efficient approach for multi-objective optimization particularly in blockchain-enabled systems. Works such as in [30] and [31] proposed the use of DRL-based optimization schemes for resource management in blockchain-based mobile edge computing (MEC) systems. The authors in [32] proposed a multi-agent reinforcement learning framework for resource management in blockchain-enabled digital twin IoT systems. The authors in [33] proposed utility optimization for blockchain-empowered edge computing using multi-agent deep deterministic policy gradient (DDPG). The authors in [34] employed a DRL-based scheme for blockchain-based resource trading and autonomous radio access network (RAN) slicing in 5G. The authors in [35] proposed a blockchain framework for healthcare systems with an intelligent blockchain manager. The blockchain manager uses DRL algorithms for the optimization of blockchain performance based on transaction characteristics.

The work of [36] explores a similar storage optimization problem to that addressed in a blockchain-IIoT setting within this work. MEC provides cloud computing resources in proximity to end devices to reduce service latency and is further supplemented by edge caching techniques [37], [38], [39]. The authors in [36] proposed a dynamic edge caching technique with privacy preservation in MEC networks to ensure efficient utilization of caching resources. The optimization of edge caching under the constraint of privacy preservation is described as a distributed optimization problem and converted into a distributed model-free MDP. They proposed a distributed DRL algorithm for preserving privacy and maximizing the cache hit rate of devices in MEC networks. They also made use of federated learning to predict content popularity. Despite all these proposed works with DRL in blockchain systems, to the best of our knowledge, DRL has not been applied to the block selection problem.

III. SYSTEM MODEL

Inspired by the works of [13] and [14], we propose the optimization of storage with a focus on individual peers. Our model presents a fully decentralized approach that allows each peer to optimize storage based on the storage resources available to the peer and the requirements of the organization operating the peer. The proposed scheme forms part of a blockchain-IIoT architecture that satisfies the following requirements:

1) The architecture should facilitate data sharing between different organizations and different IIoT networks using a blockchain network. Different organizations set access controls on their IIoT networks to maintain security and privacy. These IIoT networks can be integrated with permissioned blockchains, further extending this security but enabling better auditing and traceability for the benefit of relevant stakeholders.
2) The proposed solution should focus on the optimization of storage on individual peers. We argue that solving for the global optimum on each peer would demand a lot of computational resources and require using a central agent or server to perform the computation and relay the solution to the peers.

The system depicted in Figure 1 shows the proposed adaptive optimization scheme that is executed on the blockchain peer. Dedicated peers serving various organizations and factories on the blockchain network are linked to cloud services for block offloading. IIoT devices generate transactions within the internal networks of these organizations, which are then published to the blockchain by the linked peer. Peers in the blockchain network would execute the DRL-based optimization scheme during idle periods to send the optimal amount of blocks to the cloud.

Users submit transaction proposals and queries through a client application that communicates with the peer service. The peer service handles the transmission of transaction proposals to other peers on the blockchain and the reception of blocks from other peers. The peer service of the blockchain application commits blocks to the ledger and retrieves queried information. The peer service handles the transmission of transaction proposals to other peers on the blockchain and the reception of blocks from other peers. The DRL model interacts with the ledger and moves blocks between cloud and local storage on the local peer. It updates the peer service with the current state of the ledger before queried information can be retrieved. The block selection is made during idle time to reduce the impact of the DRL computation on other processes.

The optimization scheme that is proposed in this work and detailed in the subsequent section focuses primarily on three aspects of the problem under scrutiny, each of which correspond to an objective function in the mathematical framework. These aspects are:

1) The effect of queries due to storage location on the blockchain's performance.
2) The cost of combining cloud storage and local blockchain storage and ensuring efficient communication between both storage locations.
3) The availability of local storage by offloading blocks to cloud storage.

These objectives directly affect the performance of the blockchain peer. The conflicting nature of these objectives leads to their presentation as a multi-objective optimization problem. Although not in the scope of this paper, the security of the blockchain data stored in the cloud is another objective that can be considered. The security of cloud storage services can sometimes be called into question. Thus it may be noteworthy to determine the sensitivity of information stored on the blockchain and whether it should be stored in the cloud or not.

IV. PROBLEM FORMULATION

Certain factors need to be considered when exploring the optimization of storage in the blockchain-IIoT environment by offloading blocks to cloud storage. Traditionally, the full replication approach of blockchain technology ensures that each peer stores the entire ledger, and queries for information on the peer are handled locally. A blockchain architecture that stores the ledger partially in the cloud introduces extra query costs due to the communication between the peer and the cloud service and read/write operations in the cloud. This inevitable query cost should be optimized to ensure the query efficiency of the blockchain system is not heavily affected. In the high transaction environment of blockchain-IIoT, the data stored in older blocks may be queried less frequently as the blockchain ledger grows. The determination of the query frequency of a block and thus the likelihood of a query on that block when stored in the cloud should affect our decision to select the block to be stored in the cloud. The costs associated with cloud services should also be evaluated. The block selection problem encapsulates the impact of these considerations.

A. THE BLOCK SELECTION PROBLEM

The block selection problem is an optimization problem in which the conflicting objectives of query probability of blocks in the cloud, cloud storage cost, and local space occupancy should be minimized. This optimization problem entails getting some blocks to cloud storage in order to reduce the amount of local storage needed by the blockchain on a peer while preserving optimal blockchain performance. The objective is to select M, the ideal set of blocks to transfer to cloud storage, out of N, the total number of blocks on the blockchain. In our method, cloud and local storage blocks can be selected and moved. As illustrated in Figure IV-A4, a selected locally stored block is transferred to the cloud. Likewise, a block in the cloud that is selected is brought back to local storage.

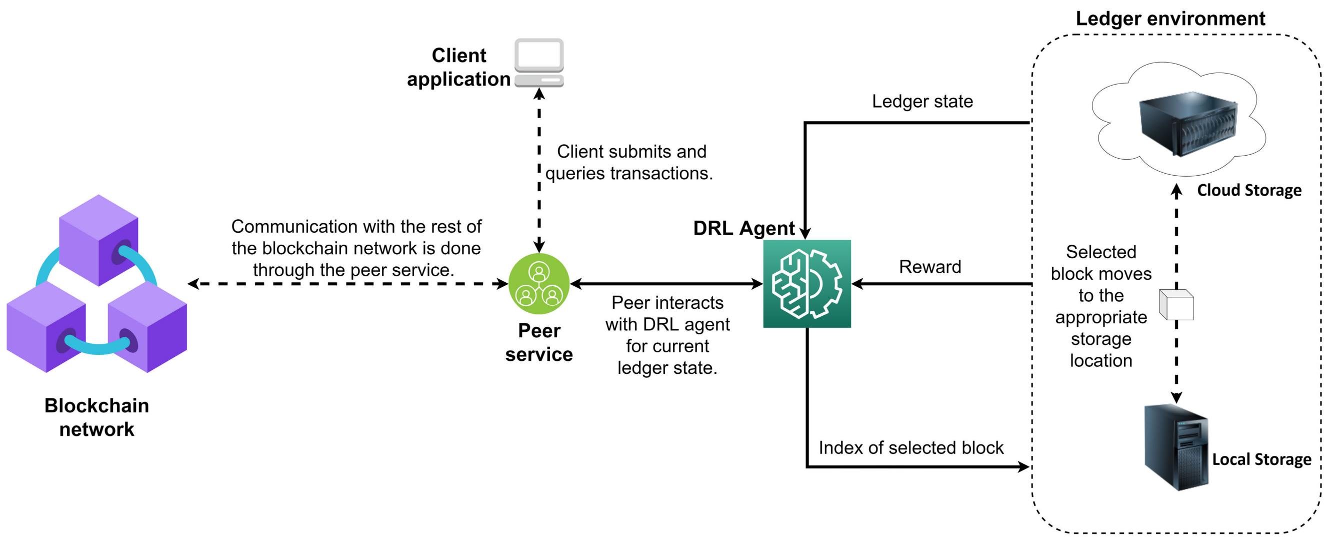
Ledger environment
Client application
Client submits and queries transactions.
Ledger state
Cloud Storage
Communication with the rest of the blockchain network is done through the peer service.
DRL Agent
Reward
Selected block moves to the the appropriate storage location
Peer service
Peer interacts with DRL agent for current ledger state.
Blockchain network
Index of selected block
Local Storage

FIGURE 1. **DRL-based storage optimization scheme.**

1) QUERY PROBABILITY

In real-world blockchain applications, the query frequency of information is in constant flux. Depending on the application under consideration, the query frequency of a block on the blockchain can be determined under three different query circumstances [13]: the fixed case, linear decay, and exponential decay.

$$F(t) = F_0, \tag{1}$$

$$F(t) = F_0 - \alpha_1 t, \tag{2}$$

$$F(t) = F_0 e^{-\alpha_2 t}. \tag{3}$$

F_0 is the initial query frequency of a block, while a_x is the attenuation coefficient. The query frequency of a block in the fixed case, given by (1), remains constant throughout the block's existence. The fixed case arises in applications for traceability. The linear decay shown by (2) is typical of situations where blocks are not regularly queried. As indicated in (3), the exponential decay most accurately characterizes financial applications like cryptocurrency networks. A block's query probability depends on the deployed application's query frequency. In this case, minimizing the query probability of blocks in cloud storage reduces the query cost as fewer cloud requests are made.

When a block is created, its time value t is set to 0. The addition of a new block raises the value of t for the preceding block by 1. At $t = 0$, a block has the initial frequency, F_0. The query probability of a block on a peer is $\{P_{b_1}, P_{b_2}, \ldots, P_{b_N}\}$ where $b_k(1 \leq k \leq N)$. The block b_N has a query probability equal to the initial query frequency. The query probability of a block is given by:

$$P_{b_k} = \begin{cases} \dfrac{1}{\int_0^{N-k} F(t)dt}, & 1 \leq k \leq N-1, \\ F_0, & k = N. \end{cases} \tag{4}$$

The total of all query probabilities for a peer's blocks is used to normalize the query probabilities. The normalized query probabilities are denoted as $P'_{b_1}, P'_{b_2}, \ldots, P'_{b_N}$. Consequently, the normalized query probability of a block is given by:

$$P'_{b_k} = \begin{cases} \dfrac{1}{P_{total} \int_0^{N-k} F(t)dt}, & 1 \leq k \leq N-1, \\ \dfrac{F_0}{P_{total}}, & k = N. \end{cases} \tag{5}$$

where $P_{total} = \sum_{k=1}^{N-k} P_{b_k} + F_0$. The overall query probability of the peer is given by:

$$P_M = \sum_{k=1}^{M} P'_{b_k}, \quad 1 \leq M \leq N. \tag{6}$$

2) CLOUD STORAGE COST

The cloud storage cost entails both the latency and monetary expense of keeping blocks in cloud storage. For industrial applications, the block size can vary and impact both the cost and latency of cloud service requests. Cloud service providers define their price structures, enabling people and businesses to select packages suited for the intended purpose. We can determine the cost of block offloading by referencing the selected price plan. While moving some blocks to the cloud is vital, it cannot be done without considering the organization's budget and the cloud requests' influence on latency. The minimization of cloud storage cost ensures that the system implementation incurs less cost in terms of money and latency.

The storage cost, C_M, depends on the monetary cost of cloud storage and the request latency for querying and transmitting a block [40]. The request latency of a block is given by:

$$l_{q_i} = (E_t \times 2) + (\frac{s_{b_k}}{B}), \tag{7}$$

where $E_t \times 2$ is the round-trip time (RTT) caused by TCP connection establishment, s_{b_k} is the size of the block and B is the network bandwidth. The monetary cost can be calculated from the pricing plan of the cloud service provider [40]:

$$M_b = (u_s \times \sum_{k}^{M} s_{b_k}) + (r_b \times u_r) + (u_t \times \sum_{k}^{M} s_{b_k}), \quad (8)$$

where M_b is the monetary cost, u_s is the unit monetary cost for storage, r_b is the total number of requests, u_r is the unit monetary cost for number of requests and u_t is the unit monetary cost for the total transmission size. The pricing used in our work is based on the Amazon S3 pricing plan. The storage cost is given by:

$$C_M = \theta l_{q_k} + M_b, \quad (9)$$

where $\theta \in [0, \infty]$ is a trade-off factor, given in dollar per second, showing the relative importance of latency and monetary cost in the storage cost function [41]. By varying θ, the storage cost ranges from low impact of latency to the highest impact.

3) LOCAL SPACE OCCUPANCY

The local space occupancy describes the local storage that the blockchain occupies. This objective is related to the fraction of blocks stored locally and the physical storage size. The higher the local space occupancy, the less storage is available for other applications on the blockchain peer. Minimizing local space occupancy increases the number of blocks transmitted to the cloud for storage and the cloud's storage charges. Due to their limited physical storage capacity, peers may prioritize this objective. Reducing local space occupancy without considering query frequency and cloud storage cost would negatively impact query performance on the peer.

The local space occupancy can be denoted by O_M and is given by:

$$O_M = \frac{e^{\frac{N_s - M}{N_s}} - 1}{e - 1}, \quad 1 \leq M \leq N, \quad (10)$$

where M is the number of blocks in cloud storage and N_s is the maximum number of blocks that can be stored on a given peer based on the available physical storage, which can be expressed as:

$$N_s = \frac{D}{s_{avg}}, \quad (11)$$

where D is the available physical storage on the peer and s_{avg} is the average size of a block.

4) MULTI-OBJECTIVE FORMULATION

The work aims to minimize all the objective functions while transferring the maximum number of blocks possible to cloud storage. The corresponding objective functions are represented as shown in (12) – (14).

$$\min \sum_{k=1}^{M} P'_{b_k}, \quad (12)$$

$$\min\ \theta l_{q_i} + M_b, \quad (13)$$

$$\min\ \frac{e^{\frac{N_s - M}{N_s}} - 1}{e - 1}, \quad (14)$$

$$\min\ (O_M, P_M, C_M),$$

$$\text{s.t. } 1 \leq M \leq N. \quad (15)$$

As shown in (15), the block selection problem is formulated as a minimization problem based on the three objective functions: the query probability, the cloud storage cost and the local space occupancy. The storage of blocks on the cloud increases the latency of transaction queries, which could impact the network's performance. Consequently, it is desirable to reduce this delay by analyzing the probability of queries on blocks and identifying which blocks would incur the least latency due to the frequency of their queries. The latency of the connection between the cloud servers and the peer and the monetary cost of using the cloud service culminates in an overall cost to storing blocks in the cloud. Reducing the portion of the blockchain held locally on each peer despite the attendant challenges of increasing query latency and storage cost is important. These conflicting objectives constitute the crux of our multi-objective optimization problem, whose minimization leads us to the ideal solution for the blockchain system.

B. BI-DIRECTIONAL BLOCK SELECTION

The approach proposed in solving the block selection problem for storage optimization is a DRL approach. The goal is to train a DRL model to select the optimal set of blocks that can be transferred to cloud storage. The next step is to present this task as a reinforcement learning problem.

1) BLOCK SELECTION PROBLEM AS AN MDP

The Markov Decision Process (MDP) forms the basis for reinforcement learning problems. The MDP offers a formal framework to describe decision making in partly stochastic and partly controlled environments. As shown in Figure 3, state transitions in an MDP must satisfy the Markov property which means the next state of the observed environment depends exclusively on the current state and action taken by an agent in that environment [29]. An MDP can be defined by the 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, which corresponds to the state space, action space, state transition function and reward function.

Concerning our block selection problem, the agent lacks knowledge of the state transition function which is the probability of transitioning from one state to another given a certain action is taken by the agent. The information used by the agent in learning is obtained through its experience of the states, actions and rewards from the environment. In this work, the block selection problem is formulated as an MDP and solved using DRL algorithms.

The system state s, at any given time step t, can be denoted by:

$$s_t = [F_t, s_t, B_t], \quad (16)$$

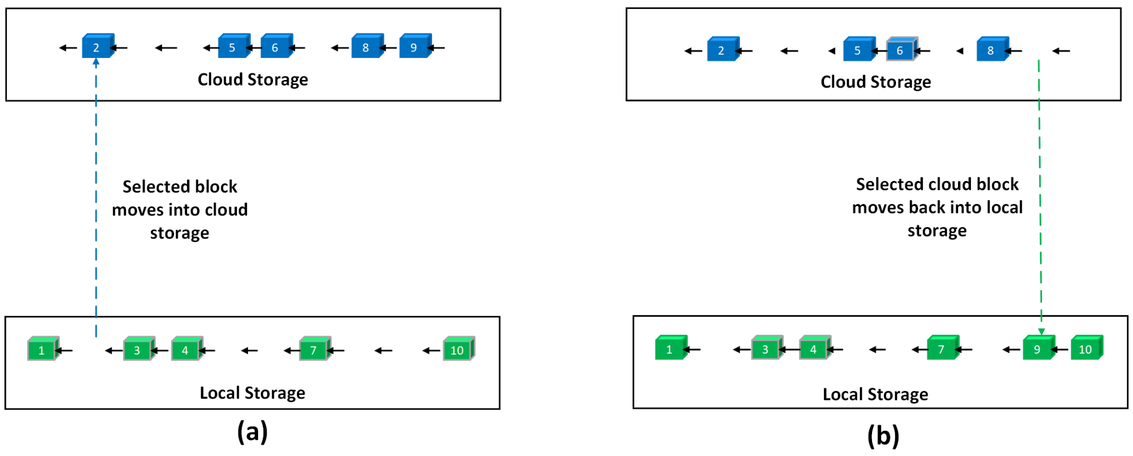
2
5
6
8
9
Cloud Storage
Selected block moves into cloud storage
1
3
4
7
10
Local Storage
(a)
Selected cloud block moves back into local storage
(b)

FIGURE 2. Bi-directional block selection: (a) The agent selects a locally stored block and moves it to the cloud. (b) The agent selects a block stored in the cloud and moves it to local storage.

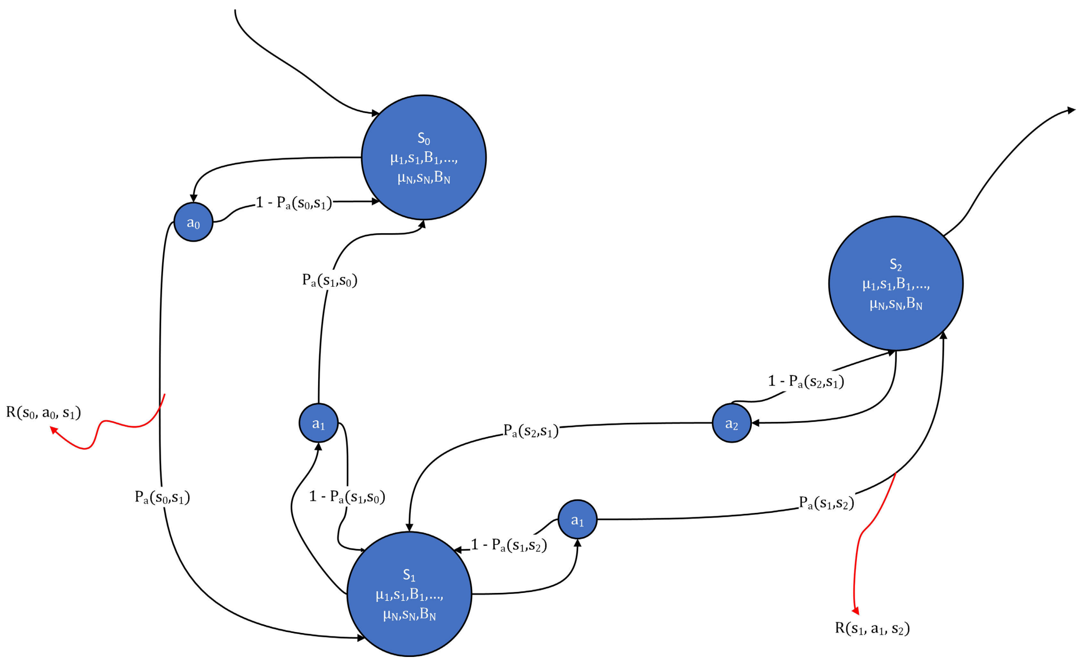

FIGURE 3. Block selection as an MDP.

where $F_t = \{F_{1_t}, \ldots, F_{N_t}\}$, $s_t = \{s_{1_t}, \ldots, s_{N_t}\}$ and $B_t = \{B_{1_t}, \ldots, B_{N_t}\}$. F_t represents the initial query frequency of a block, s_t represents the size of the block and B_t represents the block status.

The agent needs to select an action at every time step. The actions available to the agent is given by the discrete set of block numbers. The action is given by the number of the block to be selected. If no block is selected, the action is represented by zero.

$$a_t = \begin{cases} k, & 1 \le k \le N, \\ 0, & \text{no block selected.} \end{cases} \tag{17}$$

The reward function incorporates the objective functions to ensure maximum system rewards related to the optimization objective [42], [43]. The definition of the reward function expresses the prime goal of optimizing the trade-offs in the query probability, cloud storage cost and the local space occupancy whilst guaranteeing that application-level requirements such as limits on cloud cost and local storage usage are met. In contrast to the optimization problem, which seeks to minimize objective functions, reinforcement learning algorithms aim to maximize rewards. Consequently, the minimization function is transformed into maximization in the reward function. Penalties are applied to the reward under certain conditions as stipulated in (19) to provide dense rewards for the agent. The reward function is given by:

$$R = \alpha(1 - P_M) + \beta(1 - \frac{C_M}{C_{max}}) + \gamma(1 - O_M). \tag{18}$$

α, β and γ represent the objective function weights for denoting the importance of each objective relative to the other objectives and are such that $\alpha + \beta + \gamma = 1$.

$$r_t = \begin{cases} R, & D_U < D_L \quad and \quad C_M \leq C_L, \\ -R, & D_U \geq D_L \quad and \quad C_s = 0, \\ (\frac{D_L - D_U}{D_L}) \times R, & D_U \geq D_L \quad and \quad C_s > 0, \\ 0, & C_M > C_L. \end{cases} \tag{19}$$

The conditions under which the reward is provided to the agent is described as follows:

- $D_U < D_L \quad and \quad C_M \leq C_L$ D_U is the used or occupied physical storage at time t, D_L is the maximum physical storage for blockchain storage imposed by the administrator and C_L is the maximum cloud cost imposed by the administrator. The agent receives the full reward when the physical storage and cloud storage usage remain below the thresholds set.
- $D_U \geq D_L \quad and \quad C_s = 0$ C_s is the storage occupied by the blocks in the cloud. When the used physical storage goes beyond the threshold set but cloud storage remains at zero, the agent receives a negative reward.
- $D_U \geq D_L \quad and \quad C_s > 0$ When D_U goes beyond the threshold and C_s is increasing, the agent receives a negative reward whose magnitude depends on how close D_U is to D_L.
- $C_M > C_L$ If the action selected by the agent results in cloud storage cost higher than the cloud cost limit, the reward received is zero and the episode terminates.

The MDP environment represents the blocks stored in the blockchain ledger. The agent observes the ledger, selects a block and receives a reward as feedback from the environment at every time step. The episode terminates after a specified number of time steps based on the number of blocks on the blockchain. If an action a_t leads to $C_M > C_L$, the episode will terminate and an application-level alert is given to the administrator to prevent additional cloud costs. The agent is oblivious to the environment's design and discovers the optimal policy by interacting with the environment and observing the rewards for different actions in different states. The agent attempts to maximize the cumulative reward through an optimal policy. The agent learns a good policy by interacting with the environment over many episodes.

2) PROPOSED DRL ALGORITHMS

DRL employs three learnable functions: the policy, the value function, and the environment model. All reinforcement learning algorithms are based upon these functions. The policy, π, maps states to actions and dictates the actions of the agent in the environment. The value function, which has two forms, $V^{\pi}(s)$ and $Q^{\pi}(s, a)$, enables agents to evaluate the quality of accessible states and actions based on the expected future return. The environment model is represented by the transition function $P(s'|s, a)$.

Model-free reinforcement learning is considered in our proposed method because we lack a model of the environment's dynamics. Model-free reinforcement learning algorithms are not required to utilize the transition dynamics of the environment explicitly. We propose using two deep reinforcement algorithms, Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO), to evaluate the effectiveness of this approach to the block selection problem.

A2C utilizes a learned value function and a parameterized policy for learning in the environment. The actor learns a parameterized policy, while the critic learns a value function to send as feedback to the actor. In the A2C algorithm, the feedback signal is the advantage function $A^{\pi}(s, a)$, which defines how desirable or poor an action is relative to the average action that is accessible, as illustrated in (20).

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t), \tag{20}$$

The advantage function, as a relative measure, prevents actions from being punished due to the policy's current state. The advantage function does not reward actions when the policy is in a good state either. Instead, the advantage function evaluates actions based on how they change the value in the future. The variance of the learned value function is less than that of other methods, such as a Monte Carlo estimate. The introduction of the advantage function as a baseline improves the stability of the learning. As demonstrated in (21), the actor employs the policy gradient to learn a parameterized policy π_{θ}.

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_t[A_t^{\pi} \nabla_{\theta} log \pi_{\theta}(a_t|s_t)], \tag{21}$$

where $\pi_{\theta}(a_t|s_t)$ is the probability of the agent taking the action at time step t. The parameters for the actor network are updated using gradient ascent over the policy gradient:

$$\theta' \longleftarrow \theta + \nabla_{\theta} J(\pi_{\theta}), \tag{22}$$

where θ' represents the new parameters.

PPO [44] addresses the problem of performance collapse in policy gradient algorithms by adding a surrogate objective for monotonic policy improvement. The surrogate objective, shown in (23) is usually constrained by a Kullback–Leibler (KL) penalty or a clipping heuristic as shown in (24) and (25) respectively.

$$\mathcal{J}^{CPI}(\theta) = \mathbb{E}_t[r_t(\theta) A_t^{\pi_{\theta_{old}}}], \tag{23}$$

where $\mathcal{J}^{CPI}(\theta)$ is the surrogate objective and $A_t^{\pi_{\theta_{old}}}$ is the advantage computed using the old policy. $\mathcal{J}^{CPI}(\theta)$ is called the surrogate objective because it contains a ratio of the old and new policies. The superscript CPI stands for conservative policy iteration.

$$\mathcal{J}^{KLPEN}(\theta) = \mathbb{E}_t[r_t(\theta) A_t^{\pi_{\theta_{old}}} - \beta KL(\pi_{\theta}||\pi_{\theta_{old}})], \tag{24}$$

where β is the adaptive coefficient and $KL(\pi_{\theta}||\pi_{\theta_{old}})$ is the KL penalty. PPO with adaptive KL penalty converts the KL constraint $\mathbb{E}_t[KL(\pi_{\theta}(a_t|s_t)||\pi_{\theta_{old}}(a_t|s_t))]$ into an adaptive KL penalty that is subtracted from the importance-weighted

Algorithm 1: A2C

```
1  Assign the entropy regularization weight $\beta \geq 0$ ;
2  Assign the actor learning rate $\alpha_A \geq 0$ ;
3  Assign the critic learning rate $\alpha_C \geq 0$ ;
4  Initialize the actor and critic parameters $\theta_A, \theta_C$ with
     random values ;
5  for $episode = 0 \ldots max$ do
6      Gather and store data $(s_t, a_t, r_t, s_t')$ by acting in
         the environment using the current policy
7      for $t = 0 \ldots T$ do
8          Calculate predicted V-value $\tilde{V}^{\pi}(s_t)$ using the
             critic network $\theta_C$
9          Calculate the advantage $\tilde{A}^{\pi}(s_t, a_t)$ using the
             critic network $\theta_C$
10         Calculate $V^{\pi}_{tar}(s_t)$ using the critic network $\theta_C$
             and/or trajectory data
11         Optionally, calculate entropy $H_t$ of the policy
             distribution, using the actor network $\theta_A$
12         Otherwise, set $\beta = 0$
13     end
14     Calculate the value loss using MSE:
15     $L_{val}(\theta_C) = \frac{1}{T}\sum_{t=0}^{T}(\tilde{V}^{\pi}(s_t) - V^{\pi}_{tar}(s_t))^2$
16     Calculate policy loss:
17     $L_{pol}(\theta_A) =$
         $\frac{1}{T}\sum_{t=0}^{T}(-\tilde{A}^{\pi}(s_t, a_t) log\pi_{\theta_A}(a_t|s_t) - \beta H_t)$
18     Update critic parameters using SGD:
19     $\theta_C = \theta_C + \alpha_C \nabla_{\theta_C} L_{val}(\theta_C)$
20     Update actor parameters using SGD:
21     $\theta_A = \theta_A + \alpha_A \nabla_{\theta_A} L_{pol}(\theta_A)$
22 end
```

advantage resulting in the KL-penalized surrogate objective shown in (24).

$$\mathcal{J}^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t^{\pi_{\theta_{old}}}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\pi_{\theta_{old}}})], \tag{25}$$

where ϵ is the clipping neighbourhood and the term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\pi_{\theta_{old}}})$ constrains the value of $\mathcal{J}^{CLIP}$ between $(1-\epsilon)A_t$ and $(1+\epsilon)A_t$. The calculation of the KL for the KL-penalized surrogate objective can be computationally expensive. This is remedied by PPO with clipped surrogate objective which introduces a simpler modification to the surrogate objective. The clipped surrogate objective limits the probability ratio $r_t(\theta)$ of the new policy to the old policy to an ϵ-neighbourhood $[1-\epsilon, 1+\epsilon]$. The bounds on the objective $\mathcal{J}^{CLIP}$ ensures that policy updates are safe as large policy updates do not deviate beyond the neighbourhood $[1-\epsilon, 1+\epsilon]$. The clipped surrogate objective variant of PPO is preferred over the KL-penalized surrogate objective variant since it is simpler and performs better.

The DRL algorithms chosen for our implementation are two of the state-of-the-art algorithms that are widely used. They support both discrete and continuous action spaces

Algorithm 2: PPO

```
1  Assign the entropy regularization weight $\beta \geq 0$ ;
2  Assign the clipping variable $\epsilon \geq$ ;
3  Assign the number of epochs $K$ ;
4  Assign the number of actors $N$ ;
5  Assign the minibatch size $M \leq NT$ ;
6  Assign the actor learning rate $\alpha_A \geq 0$ ;
7  Assign the critic learning rate $\alpha_C \geq 0$ ;
8  Initialize the actor and critic parameters $\theta_A, \theta_C$ with
     random values ;
9  Initialize the old actor network $\theta_{A_{old}}$ ;
10 for $i = 1, 2, \ldots$ do
11     Set $\theta_{A_{old}} = \theta_A$
12     for $actor = 1, 2, \ldots, N$ do
13         Run policy $\theta_{A_{old}}$ in environment for T time
             steps and collect the trajectories
14         Compute advantages $A_1, \ldots, A_T$ using $\theta_{A_{old}}$
15         Calculate $V^{\pi}_{tar,1}, \ldots, V^{\pi}_{tar,T}$ using the critic
             network $\theta_C$ and/or trajectory data
16     end
17     Let batch with size $NT$ consist of the collected
         trajectories, advantages, and target V-values
18     for $epoch = 1, 2, \ldots, K$ do
19         for $minibatch\ m\ in\ batch$ do
20             The following are computed over the
                 whole minibatch $m$
21             Calculate $r_m(\theta_A)$
22             Calculate $\mathcal{J}^{CLIP}_m(\theta_A)$ using the advantages
                 $A_m$ from the minibatch and $r_m(\theta_A)$
23             Calculate entropies $H_m$ using the actor
                 network $\theta_A$
24             Calculate policy loss:
25             $L_{pol}(\theta_A) = \mathcal{J}^{CLIP}_m(\theta_A) - \beta H_m$
26             Calculate predicted V-value $\tilde{V}^{\pi}(s_m)$ using
                 the critic network $\theta_C$
27             Calculate value loss using the V-targets
                 from the minibatch:
28             $L_{val}(\theta_C) = MSE(\tilde{V}^{\pi}(s_m), V^{\pi}_{tar}(s_m)$
29             Update critic parameters using SGD:
30             $\theta_C = \theta_C + \alpha_C \nabla_{\theta_C} L_{val}(\theta_C)$
31             Update actor parameters using SGD:
32             $\theta_A = \theta_A + \alpha_A \nabla_{\theta_A} L_{pol}(\theta_A)$
33         end
34     end
35 end
```

and this makes them compatible with our MDP formulation. The algorithms chosen provide certain advantages that are essential in an IIoT environment. They are easy to implement, fast and computationally inexpensive.

V. PERFORMANCE EVALUATION

The proposed DRL approach is evaluated based on the rewards obtained, the objective functions, the blockchain storage reduction on the local peer and the run-time. The DRL algorithms used are compared to the evolutionary algorithms: the Advanced Time-Variant Multi-Objective Particle Swarm Optimization (AT-MOPSO) and the Non-dominated Sorting Genetic Algorithm - III (NSGA3). The curves shown in the graphs in this section were smoothed over a window of 50 episodes to provide better visualization of the trends.

A. EXPERIMENTAL SETUP AND DESIGN

The simulated environment for training was implemented in Python using OpenAI's Gym. The agents based on the proposed DRL algorithms were also implemented in Python using Stable Baselines3 which is based on PyTorch. The DRL agents used in our approach were first trained over 50,000 time steps before being evaluated.

TABLE 1. Optimization parameters.

Parameter	Value
N	200 blocks
N_s	300 blocks
D	300 MB
s_{avg}	1 MB
M_b	\$0.1766 per 100 MB
θ	0.1
D_L	150 MB
C_L	\$0.1766
α	0.5
β	0.2
γ	0.3
F_0	0.95
s_{b_i}	1 KB $< s_{b_i} \leq$ 2 MB
B_i	(0,1)

The starting parameters for the environment before training and evaluation are shown in table 1. Most of the values used were based on the earlier works in [13] and [14] to make some comparisons. The number of blocks on the blockchain when the environment starts is set at 200. The monetary cost M_b is derived from Amazon's S3 pricing but is scaled from per 1 GB to per 100 MB for easier analysis. The trade-off factor θ is set at 0.1 and can be varied depending on the value placed on latency in the cloud storage cost. The threshold for blockchain storage on the local storage is set at half the physical storage available and the monetary limit is also set at \$0.1766 which is a 100 MB maximum in the cloud. The objective function weights are set at 0.5, 0.2 and 0.3 for the query probability, cloud storage cost and local space occupancy respectively. The query probability objective is given the highest weight to ensure that blocks are not sent to the cloud despite high query costs. We consider a fixed case scenario for our query frequency for simplicity and have an initial query frequency set at 0.95. The size of a block can be variable and is generated randomly between 1 KB and 2 MB. The block status is toggled between 0 and 1 for local and cloud storage respectively. At initialization, all blocks are stored locally and thus have a block status of 0. The training and experiments in this work were performed on an Intel Core i5-10500T, 6-core, with 8 GB RAM.

B. REWARD EVALUATION

The trained DRL agents were tested on the simulated environment to evaluate their performance in terms of rewards obtained. On average, the A2C agent received higher rewards than the PPO agent. However the PPO agent showed improvement over the course of the test and received a higher reward at the end of the test compared to the A2C agent. The A2C agent started with an improvement in the earlier episodes of the test but dropped off at the end. A longer run would have seen a steady improvement in the PPO and most likely a higher average reward. The steady improvement of the PPO agent's rewards can be attributed to the ability of the algorithm to prevent large updates to the policy that could destabilize performance.

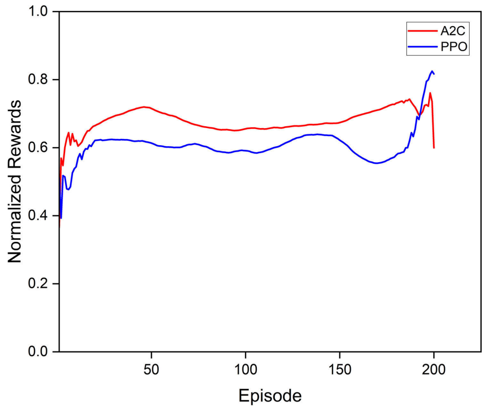

FIGURE 4. Rewards obtained by trained agents over 200 episodes.

C. PERFORMANCE ON QUERY PROBABILITY

Figure 5(a) shows the query probability at the end of each episode over 200 episodes. The results showed that the PPO agent consistently produced the highest query probability in the cloud while the NSGA3 produced the least query probability. The AT-MOPSO and A2C also had relatively low query probability. The poor performance of the PPO agent in this objective could be as result of the policy it learns over time. Since it does not allow for too large updates to the policy, the current policy used in the block selection may be over-optimistic in its selection of blocks to the cloud leading to a higher query probability.

D. PERFORMANCE ON CLOUD STORAGE COST

Figure 5(b) shows the storage cost of the DRL agents compared to AT-MOPSO and NSGA3 over 200 episodes. The values recorded are similar for the agents which are better than the AT-MOPSO and the NSGA3 in this objective. The DRL agents are more critical with regards to cloud storage costs. Since they are punished for exceeding a cloud monetary cost limit, they maintained a lower cloud storage cost hence showing better performance than the AT-MOPSO and NSGA3 in this objective.

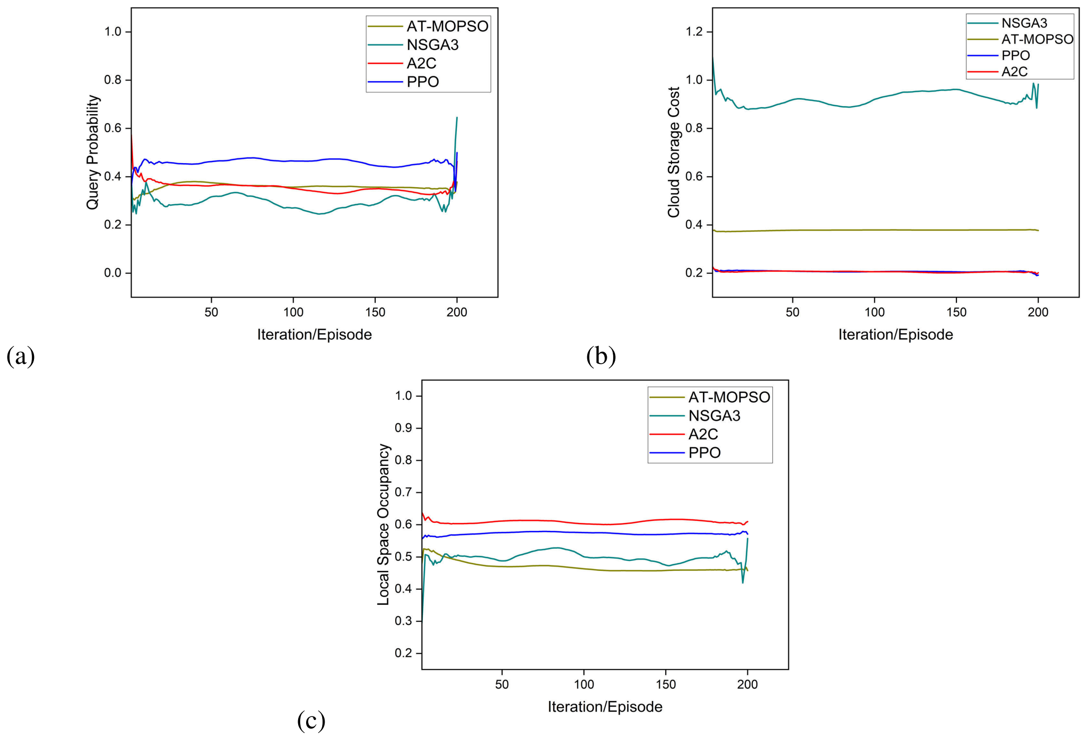

FIGURE 5. Comparison of algorithms on objective functions.

E. PERFORMANCE ON LOCAL SPACE OCCUPANCY

Figure 5(c) shows the local space occupancy of the DRL agents compared to AT-MOPSO and NSGA3 over 200 episodes. The results showed that the A2C agent maintained the highest local space occupancy and thus the worst performance in this objective. The PPO agent produced a lower local space occupancy than the A2C. The AT-MOPSO gave the best performance in this objective and had the least local space occupancy on average. The AT-MOPSO and NSGA3 produced relatively lower local space occupancy than the DRL agents. While the evolutionary algorithms gave the best performance in this objective, they are not necessarily the best choice. The evolutionary algorithm-based approach does not allow selected blocks to move back into local storage. However, in the DRL-based approach, the same block could be selected more than once and shifted between the local and cloud storage based on the observed state of the ledger and how it would affect performance. The effect of this design would be more evident in a live blockchain system over a long period of time where the performance of the blockchain can be evaluated as the query frequency of blocks change. Consequently, the average local space occupancy in our limited setup using the evolutionary algorithms is lower than the proposed approach.

F. STORAGE REDUCTION PERFORMANCE

Figure 6 shows the percentage storage reduction made using the proposed approach compared to the AT-MOPSO algorithm used in [14]. The A2C and PPO agents provided average storage reductions of 42.9% and 47.8% which are less than the NSGA3's 50.1% and the AT-MOPSO's 59.5%. The percentage storage reduction also reflected the local space occupancy objective, of which AT-MOPSO and NSGA3 performed better than the proposed approach. However, the storage saved by the DRL agents was still significant since, on average, that was nearly half of the blockchain size. Combined with their runtime, the DRL agents had an edge regarding efficiency.

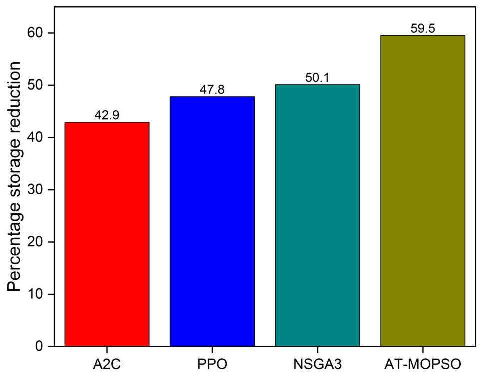

FIGURE 6. Comparison of storage reduction.

G. RUN-TIME PERFORMANCE

Table 2 shows the run-time of the trained agents over 200 iterations with 200 blocks compared to the evolutionary algorithms. The results show that the DRL approach has a much faster run-time than the evolutionary algorithms once a trained model is obtained.

TABLE 2. Run-time of DRL algorithms and evolutionary algorithm.

Algorithm	Run-time
PPO	46.7 s
A2C	53.6 s
AT-MOPSO	384.2 s
NSGA3	474.1 s

The significant difference in speed we observe between our approach and the evolutionary algorithms is because the DRL agent, once trained, only needs to examine the current state of the ledger and determine the best action to take based on its experience in training. The evolutionary algorithms generate a random set of solutions and spend time searching for the best solution in the search space for every iteration.

VI. CONCLUSION

In this paper, we examined the impact of the high storage requirements of the blockchain on its integration into IIoT systems. We proposed an adaptive optimization scheme for blockchain-IIoT systems that leverages cloud storage to reduce storage demand on local peers. To find the optimal set of blocks that should be kept in cloud storage, we proposed using a DRL agent that learns from interacting with the blockchain and evaluating the parameters of the blocks to discover which blocks should be selected. Our approach allowed the transfer of blocks in both directions as opposed to the earlier works which only looked at how blocks could be moved to the cloud. To fit as a reinforcement learning problem, we formulated the block selection problem as an MDP. We modelled the states of the blockchain environment and the actions available to the agent, and designed the reward function and reward conditions based on the objective functions for our optimization problem. We designed a blockchain simulation environment based on the MDP to train and test our models. We observed from our experimental results that storage on the local peer could be reduced by up to 47.8%. Our results also confirmed that given a trained model, our DRL approach produces solutions much faster than the classical methods used in prior works. In future works, we will explore the design of a DRL algorithm to solve the block selection problem and improve its convergence in terms of rewards. The query probability objective can also be explored in terms of temporal and spatial locality of blocks. Thus, the relationship between blocks that are linked on the blockchain could provide insight into the likelihood of queries on such blocks. It is also noteworthy to look at the integrity and security of the blockchain when blocks are transferred to and from the cloud.

REFERENCES

[1] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Dec. 11, 2021. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[3] G. Wu, S. Wang, Z. Ning, and B. Zhu, "Privacy-preserved electronic medical record exchanging and sharing: A blockchain-based smart healthcare system," *IEEE J. Biomed. Health Informat.*, vol. 26, no. 5, pp. 1917–1927, May 2022.

[4] P. P. Ray, D. Dash, K. Salah, and N. Kumar, "Blockchain for IoT-based healthcare: Background, consensus, platforms, and use cases," *IEEE Syst. J.*, vol. 15, no. 1, pp. 85–94, Mar. 2021.

[5] G. Subramanian and A. Sreekantan Thampy, "Implementation of blockchain consortium to prioritize diabetes patients' healthcare in pandemic situations," *IEEE Access*, vol. 9, pp. 162459–162475, 2021.

[6] A. Vangala, A. K. Sutrala, A. K. Das, and M. Jo, "Smart contract-based blockchain-envisioned authentication scheme for smart farming," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10792–10806, Jul. 2021.

[7] A. Vangala, A. K. Das, N. Kumar, and M. Alazab, "Smart secure sensing for IoT-based agriculture: Blockchain perspective," *IEEE Sensors J.*, vol. 21, no. 16, pp. 17591–17607, Aug. 2021.

[8] I. A. Omar, R. Jayaraman, M. S. Debe, K. Salah, I. Yaqoob, and M. Omar, "Automating procurement contracts in the healthcare supply chain using blockchain smart contracts," *IEEE Access*, vol. 9, pp. 37397–37409, 2021.

[9] G. Caldarelli and J. Ellul, "Trusted academic transcripts on the blockchain: A systematic literature review," *Appl. Sci.*, vol. 11, no. 4, p. 1842, Feb. 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/4/1842

[10] J.-H. Huh and S.-K. Kim, "Verification plan using neural algorithm blockchain smart contract for secure P2P real estate transactions," *Electronics*, vol. 9, no. 6, p. 1052, Jun. 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/6/1052

[11] M. Stefanovic, D. Przulj, S. Ristic, D. Stefanovic, and D. Nikolic, "Smart contract application for managing land administration system transactions," *IEEE Access*, vol. 10, pp. 39154–39176, 2022.

[12] N. A. Nabeeh, M. Abdel-Basset, A. Gamal, and V. Chang, "Evaluation of production of digital twins based on blockchain technology," *Electronics*, vol. 11, no. 8, p. 1268, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/8/1268

[13] M. Xu, G. Feng, Y. Ren, and X. Zhang, "On cloud storage optimization of blockchain with a clustering-based genetic algorithm," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8547–8558, Sep. 2020.

[14] C. Nartey, E. T. Tchao, J. D. Gadze, B. Yeboah-Akowuah, H. Nunoo-Mensah, D. Welte, and A. Sikora, "Blockchain-IoT peer device storage optimization using an advanced time-variant multi-objective particle swarm optimization algorithm," *EURASIP J. Wireless Commun. Netw.*, vol. 2022, no. 1, pp. 1–27, Dec. 2022, doi: 10.1186/S13638-021-02074-3.

[15] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3103–3114, Jun. 2021.

[16] N. K. Akrasi-Mensah, E. T. Tchao, A. Sikora, A. S. Agbemenu, H. Nunoo-Mensah, A.-R. Ahmed, D. Welte, and E. Keelson, "An overview of technologies for improving storage efficiency in blockchain-based IIoT applications," *Electronics*, vol. 11, no. 16, p. 2513, Aug. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/16/2513

[17] S. Qi, Y. Lu, Y. Zheng, Y. Li, and X. Chen, "CPDS: Enabling compressed and private data sharing for industrial Internet of Things over blockchain," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2376–2387, Apr. 2021.

[18] T. Kim, S. Lee, Y. Kwon, J. Noh, S. Kim, and S. Cho, "SELCOM: Selective compression scheme for lightweight nodes in blockchain system," *IEEE Access*, vol. 8, pp. 225613–225626, 2020.

[19] A. L. Spataru, C. P. Pungila, and M. Radovancovici, "A high-performance native approach to adaptive blockchain smart-contract transmission and execution," *Inf. Process. Manage.*, vol. 58, no. 4, pp. 1–15, 2021.

[20] X. Chen, S. Lin, and N. Yu, "Bitcoin blockchain compression algorithm for blank node synchronization," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2019, pp. 10–15.

[21] A. Marsalek, T. Zefferer, E. Fasllija, and D. Ziegler, "Tackling data inefficiency: Compressing the Bitcoin blockchain," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 626–633.

[22] D. Jia, J. Xin, Z. Wang, and G. Wang, "Optimized data storage method for sharding-based blockchain," *IEEE Access*, vol. 9, pp. 67890–67900, 2021.

[23] Z. Xu, S. Han, and L. Chen, "CUB, a consensus unit-based storage scheme for blockchain system," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 173–184.

[24] B. Yu, X. Li, and H. Zhao, "Virtual block group: A scalable blockchain model with partial node storage and distributed hash table," *Comput. J.*, vol. 63, no. 10, pp. 1524–1536, Oct. 2020.
[25] X. Qi, Z. Zhang, C. Jin, and A. Zhou, "A reliable storage partition for permissioned blockchain," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 1, pp. 14–27, Jan. 2021.
[26] M. Ming, R. Wang, and T. Zhang, "Evolutionary many-constraint optimization: An exploratory analysis," in *Evolutionary Multi-Criterion Optimization*, K. Deb, E. Goodman, C. A. C. Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, Eds. Cham, Switzerland: Springer, 2019, pp. 165–176.
[27] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[28] S. J. Russell, S. Russell, and P. Norvig, *Artificial Intelligence: A Modern Approach* (Pearson series in artificial intelligence). London, U.K.: Pearson, 2020. [Online]. Available: https://books.google.com.gh/books?id=koFptAEACAAJ
[29] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1659–1692, 3rd Quart., 2021.
[30] Z. Hu, H. Gao, T. Wang, D. Han, and Y. Lu, "Joint optimization for mobile edge computing-enabled blockchain systems: A deep reinforcement learning approach," *Sensors*, vol. 22, no. 9, p. 3217, Apr. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/9/3217
[31] W. Fan, W. Zhang, L. Wang, T. Liu, and G. Zhang, "Joint offloading and resource allocation in cooperative blockchain-enabled MEC system," in *Proc. ACM Turing Award Celebration Conf.-China*, Jul. 2021, pp. 136–140.
[32] D. Wang, B. Li, B. Song, Y. Liu, K. Muhammad, and X. Zhou, "Dual-driven resource management for sustainable computing in the blockchain-supported digital twin IoT," *IEEE Internet Things J.*, early access, Apr. 6, 2022, doi: 10.1109/JIOT.2022.3162714.
[33] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Utility optimization for blockchain empowered edge computing with deep reinforcement learning," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.
[34] G. O. Boateng, D. Ayepah-Mensah, D. M. Doe, A. Mohammed, G. Sun, and G. Liu, "Blockchain-enabled resource trading and deep reinforcement learning-based autonomous RAN slicing in 5G," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 216–227, Mar. 2022.
[35] A. Z. Al-Marridi, A. Mohamed, and A. Erbad, "Reinforcement learning approaches for efficient and secure blockchain-powered smart health systems," *Comput. Netw.*, vol. 197, Oct. 2021, Art. no. 108279.
[36] S. Liu, C. Zheng, Y. Huang, and T. Q. S. Quek, "Distributed reinforcement learning for privacy-preserving dynamic edge caching," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 749–760, Mar. 2022.
[37] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.
[38] M. Du, K. Wang, Y. Chen, X. Wang, and Y. Sun, "Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 62–67, Aug. 2018.
[39] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yue, and L. Fan, "A novel framework of three-hierarchical offloading optimization for MEC in industrial IoT networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5424–5434, Aug. 2020.
[40] H. Wang, H. Shen, Z. Li, and S. Tian, "GeoCol: A geo-distributed cloud storage system with low cost and latency using reinforcement learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 149–159.
[41] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2443–2457, Aug. 2016.
[42] M. Li, F. R. Yu, P. Si, W. Wu, and Y. Zhang, "Resource optimization for delay-tolerant data in blockchain-enabled IoT with edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9399–9412, Oct. 2020.
[43] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2536–2549, Dec. 2020.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

NANA KWADWO AKRASI-MENSAH received the B.Sc. degree in computer engineering from the KNUST, where he is currently pursuing the master's degree with the Department of Computer Engineering. He is also a Researcher with the Distributed IoT Platforms, Privacy and Edge-Intelligence Research (DIPPER) Laboratory. His current research interests include machine learning and blockchain technology.

ANDREW SELASI AGBEMENU (Member, IEEE) was born in Accra, in 1981. He was a Visiting Scholar with the Analog and Mixed Signal Center, TAMU, College Station, TX, USA. He is currently a Lecturer with the Department of Computer Engineering, KNUST, Kumasi, Ghana. He is also the lead of the TI Microelectronics Laboratory, KNUST, and a Researcher with the Responsible AI Laboratory (RAIL, KNUST). His current research interests include data converters and using AI analog design. He is a member of IEEECASS Society and IEEE-SSCS Society.

HENRY NUNOO-MENSAH (Member, IEEE) received the B.Sc., M.Phil., and Ph.D. degrees in computer engineering from the Kwame Nkrumah University of Science and Technology (KNUST), Kumasi. He is currently a Professional Engineer and a Lecturer with the Department of Computer Engineering, KNUST, where he is also the Programs Coordinator for the IDRC-GIZ-funded Responsible AI Laboratory, KNUST. He is also the M.Sc. Programs Coordinator for the KNUST Engineering Education Project (KEEP), a World Bank-sponsored African Centre of Excellence for developmental impact. He is also the Projects Coordinator with the Connected Devices (CoDe) Laboratory, Department of Computer Engineering, KNUST, and a Research Fellow with the Brew Hammond Energy Centre (TBHEC), KNUST. His research outputs are published in numerous reputable journals and conferences. His research interests include wireless sensor networks, intelligent agents (AI4D), blockchain, and the Internet of Things.

ERIC TUTU TCHAO (Member, IEEE) received the Ph.D. degree in telecommunications engineering from the Kwame Nkrumah University of Science and Technology (KNUST), Ghana. He worked on analysis of MIMO antenna configuration effects on WiMAX networks deployments at KNUST, where he is currently a Senior Lecturer with the Department of Computer Engineering. He is interest in globalization and technology transfer has resulted in assisting the development of undergraduate and postgraduate programs with the Computer Engineering Department. His research interests include optimizations in blockchain, the Internet of Things networks, and AI4D. He is a member of the IEEE Communications Society and The Internet Society. He is also a member of the IETF's Special Working Group on IP over IEEE 802.16 (WiMAX) Networks and the Partnership Coordinator for the Responsible Artificial Intelligence Laboratory (RAIL). He is a DFG Scholar, a Postdoctoral Research Fellow of the World Academy of Science (TWAS) and a Future Africa Research Leadership Fellow.

ABDUL-RAHMAN AHMED (Member, IEEE) was born in Yeji, Ghana, in 1977. He received the M.Sc. degree from the University of Hull, Hull, U.K., in 2005, and the Ph.D. degree in radio science and engineering from Chungnam National University, Daejeon, South Korea, in 2015. In 2005, he joined the Department of Electrical Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, as a Lecturer. His current research interests include noise RF and microwave circuit design and noise parameters measurement techniques.

ELIEL KEELSON was born in Tema, Ghana, in 1987. He received the Ph.D. degree in computer engineering from the Kwame Nkrumah University of Science and Technology, where he is currently pursuing the Ph.D. degree. He worked on a low-cost early adoption strategy for implementing secured smart energy metering systems in African developing countries. His current research interests include artificial intelligence for development in Africa and blockchain applications in the IIoT ecosystems.

AXEL SIKORA received the master's (Dipl.-Ing., corresponding to M.Sc.) degree in electrical engineering and the master's (Dipl. Wirt-Ing., corresponding to M.B.A.) degree in business administration from Aachen Technical University, Germany, and the Ph.D. (Dr.-Ing.) degree in electrical engineering from the Fraunhofer Institute of Microelectronics Circuits and Systems, Duisburg, with a thesis on SOI-technologies. From 1990 to 1991, he was a DAAD Alumnus with the St. Petersburg Politechnical Institute. After various positions in the telecommunications and semiconductor industry, he became a Professor with Baden-Wuerttemberg Cooperative State University Loerrach, in 1999. In 2011, he joined the Offenburg University of Applied Sciences, where he currently leads the Institute of Reliable Embedded Systems and Communication Electronics (ivESK). Since 2016, he has been a Deputy Member of the Board to the Hahn-Schickard Association of Applied Research, a government-funded research institute in the state of Baden-Wuerttemberg, where he leads the engineering divisions and "software solutions." He is the author, coauthor, editor, and coeditor of several textbooks and more than 200 peer-reviewed papers in the field of embedded design and wireless and wired networking. Amongst many other duties, he serves as the Chairperson of the Annual Embedded World Conference and the world's largest event on the topic.

DOMINIK WELTE received the M.Sc. degree in computer science. He is currently a Researcher with the Institute of Reliable Embedded Systems and Communication Electronics (ivESK), Offenburg University of Applied Sciences. His research interests include secure communications, blockchain, and time sensitive networking.

JERRY JOHN KPONYO (Member, IEEE) is currently the Scientific Director of the Responsible Artificial Intelligence Laboratory (RAIL), the Lead of the KNUST Engineering Education Project, and a Co-Founder of the Responsible AI Network Africa, and a collaborative initiative with Technical University of Munich and a Visiting Professor of ESIGELEC France. He is also the Dean of the Quality Assurance and Planning Office, KNUST, under the Vice-Chancellor's Office. He is also the Former Dean of the Faculty of Electrical and Computer Engineering. He is also an Associate Professor with the Department of Telecommunications Engineering, KNUST.