EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Blockchain-IoT peer device storage optimization using an advanced time-variant multi-objective particle swarm optimization algorithm

Clement Nartey[1,2], Eric Tutu Tchao[2*], James Dzisi Gadze[3], Bright Yeboah-Akowuah[2], Henry Nunoo-Mensah[2], Dominik Welte[4] and Axel Sikora[4]

*Correspondence: ettchao.coe@knust.edu.gh
[2] Department of Computer Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
Full list of author information is available at the end of the article

Abstract
The integration of Internet of Things devices onto the Blockchain implies an increase in the transactions that occur on the Blockchain, thus increasing the storage requirements. A solution approach is to leverage cloud resources for storing blocks within the chain. The paper, therefore, proposes two solutions to this problem. The first being an improved hybrid architecture design which uses containerization to create a side chain on a fog node for the devices connected to it and an Advanced Time-variant Multi-objective Particle Swarm Optimization Algorithm (AT-MOPSO) for determining the optimal number of blocks that should be transferred to the cloud for storage. This algorithm uses time-variant weights for the velocity of the particle swarm optimization and the non-dominated sorting and mutation schemes from NSGA-III. The proposed algorithm was compared with results from the original MOPSO algorithm, the Strength Pareto Evolutionary Algorithm (SPEA-II), and the Pareto Envelope-based Selection Algorithm with region-based selection (PESA-II), and NSGA-III. The proposed AT-MOPSO showed better results than the aforementioned MOPSO algorithms in cloud storage cost and query probability optimization. Importantly, AT-MOPSO achieved 52% energy efficiency compared to NSGA-III. To show how this algorithm can be applied to a real-world Blockchain system, the BISS industrial Blockchain architecture was adapted and modified to show how the AT-MOPSO can be used with existing Blockchain systems and the benefits it provides.

Keywords: Blockchain, Internet of Things, Particle swarm optimization, Cloud storage, Scalability

1 Introduction

Blockchain has gained tremendous traction over the past decade due to its remarkable contribution to cryptocurrency. The emergence of Blockchain as a distributed ledger technology led to its application in areas such as healthcare [1, 2], supply chain management [3], education [4], real estate [5] as well as Internet of Things (IoT) [6].

Blockchain varies from centralized digital databases and ledgers because it harnesses the concept of community validation to synchronize the entries that go into the ledger. It further works via a distributed model to replicate the updated ledger to all the nodes and users involved in the network [7]. Blockchain is a very robust technology due to its decentralized, transparent, secure, immutable and automated characteristics. However, many researchers have highlighted scalability as a crucial concern affecting Blockchains and this needs critical attention [8]. The scalability setback, which includes low throughput, resource-intensive computations, and high latency, has dramatically hindered practical Blockchain-based applications.

Another critical concern that affects Blockchain adoption is the storage space requirements needed to run a Blockchain node. The space requirement is a Blockchain component that increases daily due to the Blockchain ledger's append-only nature [9]. The more transactions completed, the larger the ledger size. According to a study by Statista [10], it was reported that the Blockchain size of Bitcoin as of February 2021 stood at 321.32 GB, and it will take a new full node, a bootstrap time of roughly four days, to be a part of the network. Ethereum [11], which is also another popular Blockchain, suffers from a similar situation.

Several solutions have been proposed to handle the issue of scalability on Blockchains. Some of these solutions include sharding (on-chain scaling) [12], state channels (off-chain scaling), side chains with schemes as Plasma [13] for the Ethereum network, the bloXroute technology [14] as well as Directed Acyclic Graph (DAG) [15].

These outlined issues about Blockchains become even more profound when IoTs are integrated into Blockchains. IoT solutions require that many transactions be executed at every given time [16], but some Blockchain networks such as Bitcoin can only be capable of performing 7–20 transactions per second [8]. IoT devices are limited when it comes to the aspects of their storage and computational resources. On the other hand, rapid improvement in computing technologies has led to the development of concepts and systems such as Edge computing [17]. Edge computing is an extension of cloud computing [18], because it enables many devices to run applications at the edge of a network. It is done by providing compute resources for devices to offload tasks that are computationally intensive and provide data storage, maintain low latency, support heterogeneity and improve the quality of service for applications that require low latency, such as IoT applications [17]. Edge computing has been partly used to solve the scalability for Blockchain-IoT (BIoT) applications such that the IoT devices do not act as nodes on the Blockchain. Instead, they connect to these edge computers instead of nodes on the Blockchain [19]. These edge/fog computing structures tend to have more computational and storage resources, but the number of transactions produced by these IoT devices still puts considerable pressure on them.

Some existing solutions [20] to this storage problem mainly include storing some of the blocks produced by these edge nodes in the cloud. Most of the proposed solutions based on this have been backed by the fact that there has been significant improvement in the security and encryption schemes implemented on cloud platforms. To the best of our knowledge, not much literature exists which addresses the cloud storage problem by using optimization techniques. Xu et al. [21] formulated this problem into a multi-objective block selection problem and solved it using a Non-dominated Sorting Genetic

Algorithm with Clustering (NSGA-C). Their approach had a few limitations, which this paper seeks to improve. First of all, when compared to NSGA-III, NSGA-C performed worse in 4 out of 5 objective functions and only out performed NSGA-III in the last objective function. Secondly, the NSGA-III has a quite long runtime and NSGA-C has an even longer runtime. This is not quite ideal because for just three peers the block selection algorithm took upwards of 10 min to run. This also means that the energy consumed during the running of this algorithm was also increased. When conducting this research, we set out with the quest to have an optimization scheme that performed better for all objective functions and also run in less than the current NSGA-C algorithm. In our investigations, we also realized that no research had been performed where MOPSO had been applied to this problem, thus, we propose this novel Advanced Time-Variant Multi-Objective Particle Swarm Optimization (AT-MOPSO).

The paper aims to take advantage of the improvements made in cloud technologies and leverage the colossal storage availability on cloud systems to solve the storage problems faced by IoT nodes. We seek to accomplish this by storing some blocks in the cloud and maintaining the Blockchain's decentralized nature by leveraging different cloud providers and cloud deployment strategies such as containerization. The paper adapts our improved hybrid IoT architecture design [22] to show how our Advanced Time-Variant Multi-Objective Particle Swarm Optimization (AT-MOPSO) algorithm would help solve the block selection cloud storage problem. We also applied it to the BISS architecture [23] from one of our recent research works to improve it and further show how legacy machines with limited capabilities can be brought closer to the Blockchain as nodes. The main contributions of the paper are:

1. Proposing a framework to ensure improved operation for fog/edge nodes by leveraging the power of side-chains and containers based on an improved hybrid architecture design in [22].
2. Formulating the problem into an optimization scheme for cloud providers' storage cost, the query probabilities of blocks on IoT fog peers, and storage availability on IoT fog nodes.
3. Proposing an Advanced Time-variant Multi-Objective Particle Swarm Optimization (AT-MOPSO) Algorithm to obtain the correct number of blocks to be selected from the fog nodes for cloud storage.
4. Comparing the energy efficiency of some Multi-Objective optimization schemes against our proposed algorithm to show our scheme's energy-saving and efficient nature.
5. Incorporating our scheme into an industrial Blockchain Architecture to show our scheme can be used in a real-world scenario.

Our study primarily focuses on how the storage space on Blockchain peers be optimized by sending some of the stored blocks to a cloud storage and does not deal directly with aspects of the network or the actual connection between devices.

The rest of this paper is organized as follows. Section 2 presents an overview of Blockchain, edge computing, and multi-objective optimization techniques. Section 3 looks at the proposed improvement to fog nodes and the mathematical formulation of the multi-objective block selection problem. Section 4 outlines the details of our proposed Advanced Time-Variant Multi-Objective Particle Swarm Optimization (AT-MOPSO) algorithm and the results of our scheme in comparison to other multi-objective optimization scheme is shown in Sect. 5 for all objective functions as well as the energy-saving benefits. Section 6 shows a real world application of the AT-MOPSO algorithm and Sect. 7 concludes the paper and provides recommendations for future work.

2 Related work

2.1 Blockchain

To better understand the structure and nature of Blockchain systems, it is good to look at the decomposed layers of Blockchains, as shown in Fig. 1. The figure consists of the data, network consensus, ledger topology, and contract and application layers [20].

The data layer performs the function of data encapsulation so that the data generated from a Blockchain application or transaction is verified and hidden in a block. Such a block is linked to the previous block by its header with a hash value. The process results in an ordered chain of blocks replicated among all nodes on the Blockchain [12]. The replication process occurs in the network layer, whereby the generated blocks are propagated to all nodes in the Blockchain [24]. Since the Blockchain is seen as a decentralized

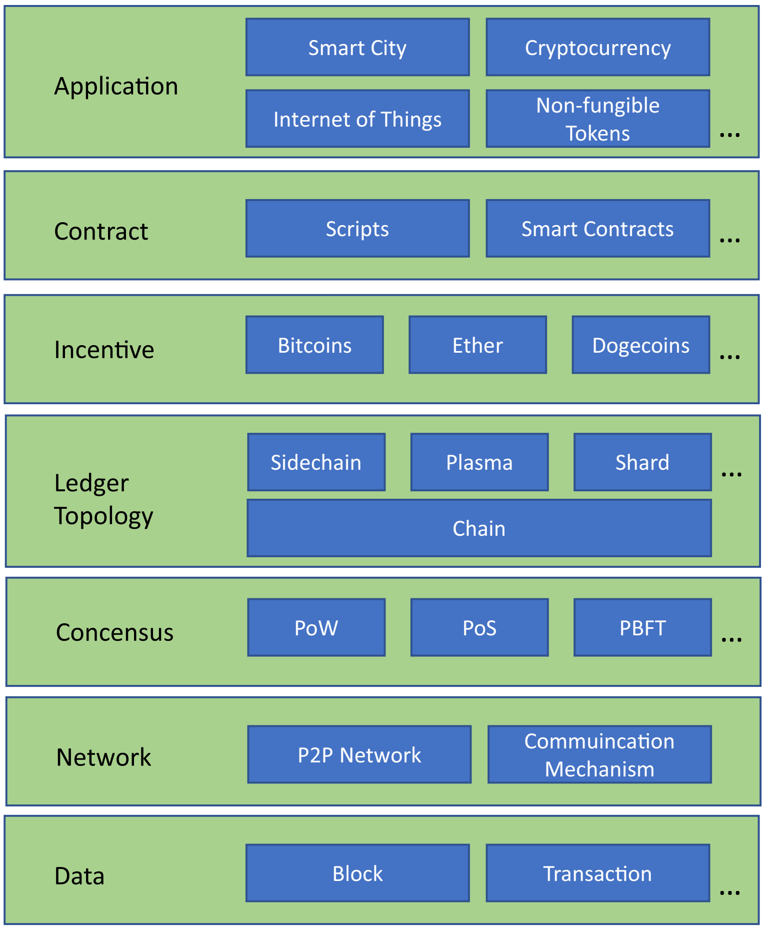

Fig. 1 The decomposition of the structure of Blockchain

network of nodes, it is modeled as a P2P network where peers act as participants and provide storage for the distributed ledger of blocks.

Consensus algorithms and schemes help maintain the integrity of the data on the Blockchain. The nodes on the Blockchain verify each propagated data block and ensure that it is a valid block before added to all the ledgers on all nodes. The three primary consensus schemes that have seen widespread usage include Proof-of-Work (PoW) [10], Proof-of-Stake (PoS) [25, 26], and Practical Byzantine Fault Tolerance (PBFT) [27].

The last decade has also seen an increase in the research and development of cloud-based architectures and technology. The high availability of computing resources, storage, and highly reliable and performance network infrastructure has led to more applications being run on the cloud [28]. The trend also led to a second wave that seeks to move these highly performance systems from the cloud to the edge of networks [20]. The shift to the edge is mainly being done to support applications that require little to no delay in their operations. Such applications include IoT implementations [17, 29] and Virtual Reality applications [30], etc. The adoption of edge/fog computing to support Blockchain and IoT applications' integration is due to the resource limitations on IoT devices and the low latency communication usually required in their operations.

We realized that much work had not been done concerning the ledger topology layer of the Blockchain system from the research that we have conducted. This is essential because this layer is required to store the authenticated blocks produced from the consensus layer. Large storage capacities are needed for this. The issue is very profound when Blockchain-IoT integrations and applications are considered, mainly because of the storage limitations on these IoT devices. Fog Computing structures used for IoT implementations take a lot of this load off IoT devices, but the large amounts of data produced by the IoT devices can still put high demands on these fog nodes.

Palai et al. [31] proposed an approach where a block would summarize the transactions of several consecutive blocks. Then this summary block would be used to make a net change on those blocks, thereby reducing the storage footprint. The only problem with this approach is that if only a few blocks are summarized, then the summary blocks' size is not any smaller than a set of continuous blocks. The authors in [32] also proposed architecture to solve Blockchains' storage problems using a class of erasure codes known as 'Fountain codes'. This architecture enables a full node on the Blockchain to encode blocks that other nodes have validated into a more compressed structure made up of different blocks, reducing the storage space needed.

Yang et al. [33] proposed a dual storage solution that uses both an on-chain and off-chain approach. Their approach was used for a fruit and vegetable traceability application with IoT devices serving as input devices. The public information about products was stored in a relational database, and the private information about the products was sent over to the Blockchain. This approach is also efficient since fewer data would be sent over to the Blockchain. The downside of such an approach is that if there is data loss on the side relational database, that data cannot be retrieved, and the private information sent to the Blockchain would be without context.

Kumari et al. [34] also proposed an off-chain approach intended for use in an innovative city application for IoT devices acting as smart meters and sensors. The off-chain solution uses the InterPlanetary File System (IPFS) [34], a peer-to-peer hypermedia protocol to store only transactions related to the devices off-chain and links the off-chain records to the main Ethereum Blockchain, thereby reducing storage requirements and costs.

2.2 Containerization

Another trend that has caught on with the popularization of cloud and edge/fog computing is the concept of containers. Fog computing, which is known to solve the latency issue for IoT devices, works by deploying a set of distributed servers and compute resources at the edge of the network. The infrastructure helps avoid data transmission over long distances, thus avoiding propagation and processing delays [29]. Complex configurations and different software environments have to be set up on fog servers to fulfill the demands of various applications' needs. Containerization has been used in such cases to help solve the problem of software deployment and configurations and migration activities [19]. The process is achieved by bundling all the relevant source codes with their respective library requirements into encapsulations known as containers which can be easily deployed on the fog servers.

Some researchers have proposed systems to use containerization orchestrations with Blockchain and IoT applications, but not much literature exists on this topic. Cui et al. [6] proposed a Blockchain-based containerization scheme for an aspect of IoTs known as the Internet of Vehicles (IoV). Their implementations focused on the container scheduling policies for Directed Acyclic Graphs (DAGs), which determined how many containers should be running to effectively manage the resources on the fog servers that the vehicles were utilizing. In [35], the authors proposed a Blockchain system based on Distributed Hash Table (DHT) called LightChain. The Blockchain implementation was deployed on a single machine using a docker container, and this was done to show its lightweight nature. The individual nodes of the Blockchain were run as separate threads in the container.

2.3 Multi-objective optimization

Multi-objective optimization is defined as the process of obtaining a suitable vector of variables from a feasible region that a set of constraints can defines. The goal is to find the vector of variables such that a vector of objective functions would be minimized or maximized. It can be expressed as follows:

$$\min F(x) = \{f1(x), f2(x), \ldots, fm(x)\}, \quad \text{s.t.} \quad g(x) \leq 0 \tag{1}$$

In this case, x represents the vector of design variables—each value of f_i, for $i(1 \leq i \leq m)$ representing an objective function that must be optimized and $g(x)$ is the constraint vector with m being the number of objective functions in question.

Evolutionary algorithms have been used widely in research to solve multi-objective problems, and the general term used for these is Multi-Objective Evolutionary Algorithms (MOEA) [36]. Such algorithms possess parallel efficiency, robustness, and versatility when applied to complex optimization scenarios. Ridha et al. [37] used multi-objective optimization to solve the problem of standalone photovoltaic system design. Their research aimed to optimize the output power obtained from a given system using storage battery capacity, mathematical models of the different types of PV modules, and environmental criteria. Cai et al. [24] also proposed a sharding scheme based on multi-objective optimization. The scheme presented an optimized shard validation model, which improved the throughput of the Blockchain network and reduced the aggregation of malicious nodes in the assigned shards.

Not much research exists where multi-objective optimization has been used to solve the block storage problem for IoT devices. Xu et al. [21] proposed an algorithm based on a Non-dominated Sorting Genetic Algorithm (NSGA-III) with K-Means clustering to obtain a crowding distance that helps to select the Pareto optimal solution from the set of generated keys. They modeled the factors needed to choose the appropriate number of blocks to be sent to the cloud. These include the local space occupancy of the IoT, query probabilities of the peers (IoT devices), and the cloud's storage cost.

In this paper, we propose an Advanced Time-variant Multi-Objective Particle Swarm Optimization Algorithm (AT-MOPSO), which is based on the Multi-Objective Particle Swarm Optimization (MOPSO) [38] algorithm. MOPSO is a variant of Particle Swarm Optimization (PSO) for solving multi-objective optimization problems. Both PSO and MOPSO have a defect: they can get stuck in the local optimal solution space in the search process for the optimal global solution. The AT-MOPSO seeks to solve the MOPSO algorithm's defects and provide results for the optimal number of blocks that should be sent to the cloud for each IoT device.

3 System model and problem statement

3.1 System model

In this paper, we propose a system for fog servers specifically for Blockchain-IoT applications such that a side-chain would be run on the fog server, and the nodes of the side-chain would be individual containers. Each of these containers would serve an IoT device and offload all Blockchain-related activities from the device. The fog node would itself serve as a full node on the main Blockchain.

In our proposed IoT-Fog node architecture in Fig. 2, the Fog computing node would not just be a node connected to the Blockchain. The Fog node would instead run a side chain for the devices that are connected to it. In such a case, completed transactions on the side chain will be sent to the main chain. The Blockchain mining process and interactions are still going to be abstracted from the IoT peer devices. The fog will

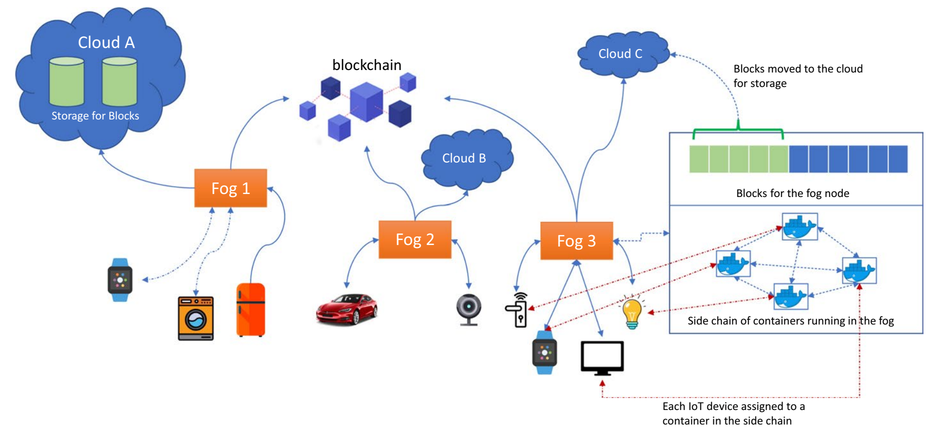
Cloud A
Storage for Blocks
blockchain
Cloud C
Blocks moved to the cloud for storage
Cloud B
Fog 1
Fog 2
Fog 3
Blocks for the fog node
Side chain of containers running in the fog
Each IoT device assigned to a container in the side chain

Fig. 2 Proposed system for running a side chain on a fog node using containers and microservice

not employ a traditional service-oriented monolithic architecture [39], and the nodes of the Blockchain would not be the IoT peer devices. The way the side chain would operate would be by using microservices [6]. Thus, multiple microservices would run as containers on the fog that would act as nodes on the side-chain. These containers would take up the mining activities from the IoT devices.

Each IoT device would be randomly assigned to a Node in the side-chain (i.e., a container). Separate containers would not be created for each IoT device, but rather a pool of IoT devices would be assigned to a node at a time. The depiction of this proposed update is shown in Fig. 2.

Only a limited amount of delay is introduced for message propagation through the side-chain, ensuring high throughput of transactions resulting from the small number of container nodes. This architecture would be best suited for the high transaction—high-performance IoT implementations.

As mentioned earlier, the Blockchain's storage requirements on the fog node are quite intensive because even a low throughput Blockchain such as Bitcoin has a ledger size of about 321.21 GB. In contrast, a high throughput Blockchain such as Ripple has a ledger size of up to 9TB [32]. Our proposed system curtails this challenge by moving blocks from fog nodes to the cloud for storage, as shown in Fig. 2.

3.2 Problem statement

Based on our proposed model, the question that arises is which blocks should be moved to the cloud and how many of those blocks should be moved to ensure the smooth and efficient running of the IoT devices connected to it. This problem is formulated with the appropriate mathematical models are presented here in this section of the paper.

Suppose we have several fog servers acting as fog nodes or peers for IoT devices connected to it as $S = \{s_1, s_2, \ldots, s_m\}$, where $s_i \in S$ denotes a single server and m the total number of fog nodes being considered. For each fog node/peer, we need to select some blocks to be sent to the cloud to alleviate that peer's storage pressure. The blocks in each fog node can be represented by $b_1, b_2, \ldots, b_N$, where N is the number of blocks mined by the peer at any given time.

The number of blocks that would be taken to the cloud can be represented by $M_\mathrm{w}(1 \le M_\mathrm{w} \le N)$, where $w = s_i \in S$ as shown in Fig. 3.

Once M_w blocks are sent to the cloud the blocks in the fog peer are renumbered such that $b_{M_\mathrm{w}+1}$ then becomes b_1 and so on.

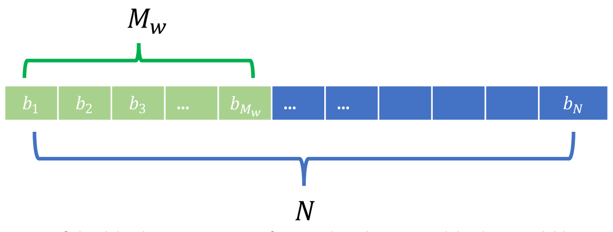

Fig. 3 A representation of the blocks present in a fog node where *Mw* blocks would be transferred

Based on the type of application that the Blockchain-IoT implementation is being used to serve, we can have three different query conditions for the blocks on the fog node. The query conditions can be for either a fixed case or linear decay scenarios. The fixed case scenario when used in a traceability application such as [33] denoted in (2). The linear decay scenario describes when the blocks are not queried as often as denoted by (3). In exponential decay scenarios, the Blockchain is used as storage for transactions such as in cryptocurrencies, as denoted by (4).

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

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

$$F(t) = F_0 e^{-\alpha_1 t} \tag{4}$$

where α_1 represents the attenuation coefficient in the exponential decay scenario, and α_2 represents that for the linear decay scenario. These can be determined based on the Blockchain use case scenario.

Xu et al. [21] proposed that the number of blocks taken to the cloud for each peer on a Blockchain can be formulated into a multi-objective optimization problem based on three main objective functions. These are the Query Probability of the various blocks on the fog node. The cloud storage cost when some blocks are moved to the cloud, and the local space occupancy represents that amount of local space on the fog node that will be alleviated by sending some of the blocks to the cloud. The mathematical models for these objective functions are outlined as follows:

3.2.1 Query probability

The query probability for the blocks in a fog node is based on the query frequency $F(t)$ for the type of Blockchain-IoT application being implemented. The value represented as t for every block is tightly coupled to when that block was generated. Thus, with the addition of every new block, t is increased by 1. This means that the first generated block in the set has a t value of 0 as shown in (5). In the eventual scheme of events, the first generated block would be the last block in the arrangement given by b_N as shown in Fig. 3.

$$\Rightarrow t = 0, F(t) = F_0. \tag{5}$$

The query probability for the blocks in a fog node can be represented by $P_{b_1}, P_{b_2}, P_{b_2}, \ldots, P_{M_w}, \ldots P_{b_N}$ where $b_j\left(1 \le j \le N-1\right)$. Thus, the query probability for the various blocks can be found by (6). It must be noted the block b_N would have both the query frequency and query probability of F_0 since it was the first block created.

$$P_{b_j} = \begin{cases} \left[\frac{1}{\int_0^{N-j} F(t)\mathrm{d}t}\right] & 1 \le j \le N-1 \\ F_0 & j = N \end{cases}. \tag{6}$$

Based on (6), we can calculate the sum of the all the query probability for all the blocks for a fog node as Λ as shown in (7). The sum can be used to normalize the values of the query probabilities represented by $P'_{b_1}, P'_{b_2}, P'_{b_2}, \ldots, P'_{M_w}, \ldots, P'_{b_N}$ as shown in (8).

$$\Lambda = \sum_{j=1}^{N-j} P_{b_j} + F_0. \tag{7}$$

$$P'_{b_j} = \begin{cases} \left[\frac{1}{\Lambda \int_0^{N-j} F(t)\mathrm{d}t}\right] & 1 \leq j \leq N-1 \\ \frac{F_0}{\Lambda} & j = N \end{cases}. \tag{8}$$

Thus, after all the query probabilities of the blocks have been found, the overall query probability for the fog node is based on the number of blocks M_w to be sent to the cloud can be found. This can be achieved by finding the sum of all the normalized query probabilities up to the M_wth block. For fog node s_i, the overall query probability is denoted by P_{s_i} as shown in (9).

$$P_{s_i} = \sum_{j=1}^{M_w} P'_{b_j} \quad 1 \leq s_i \leq m. \tag{9}$$

where $w = s_i$ and $m = |S|$.

Based on the value of m, there will m objective functions; one for each fog node s_i with P_{s_i} being minimized.

3.2.2 Storage cost

The storage cost deals with the cost of storing the blocks in the cloud. The storage cost for cloud storage is assumed to be the same for all the fog nodes for the sake of simplicity. The size of one block for each fog node is represented by C. Different Blockchain has different sizes for the blocks that are generated. Thus, the Blockchain being used must be considered, and the block size of an individual block must be known. For example, the bitcoin Blockchain is known to have blocks with a size of 1 MB [40], the size of blocks on the Hyperledger Fabric Blockchain can be adjusted to be as large as possible [41]. The size of a block on the Ethereum Blockchain varies based on the gas limit [11].

The storage cost is considered a linear function as shown in (10), which is the total size of all blocks moving from the fog node to the cloud. This linear function is governed by a factor k representing the ratio of the cost of cloud storage compared to local storage. Thus, when k has a small value, it means that cloud storage is cheaper than local storage options for the fog node and vice versa. This value is based on the cloud service provider used by a fog node and the type of local storage available on the fog node, such as Optical Hard Disk Drives (HDD)—which are slightly cheap but slow, or Solid-State Drives (SSD)—which are relatively more expensive and faster.

$$cost = kC \times \sum_{w=1}^{m} M_\mathrm{w}. \tag{10}$$

where $m = |S|$.

3.2.3 Local space occupancy/storage availability of IoT Fog node

Local space occupancy/the storage availability of IoT Fog nodes is directly related to the number of stored blocks in the cloud. This is because the more blocks stored in the cloud, the more space would be made available on the fog node, thereby easing the given node's storage pressure. The local space occupancy is inversely proportional to the storage availability of a node. More blocks sent over to the cloud for storage means local storage occupancy would be small, and thus there would be an increase in the storage available on the IoT fog node. It should be noted that the storage size on each fog node is determined by the number of IoT devices connected to it, the type of application that it is being used for, and the operator of that fog node. Thus, some nodes may prioritize local space occupancy/storage availability more than others. In this sense, weights can be assigned to each fog node such that nodes that prioritize storage availability and have low storage capacity would be given bigger weights. The weights can be represented as β_{s_i}, such that $\beta_{s_1}, \beta_{s_2}, \beta_{s_3}, \ldots, \beta_{s_m}$ represent weights for individual fog node where $m = |S|$. The weights would be used in a weighted sum equation to find the overall local space occupancy for the fog nodes; thus, they would be given decimal values which would sum up to 1 as shown in (11).

$$\sum_{i=1}^{m} \beta_{s_i} = 1 (1 \leq i \leq m). \tag{11}$$

The individual local space occupancy for each fog node can be denoted by Q_{s_i}, expressed in (12). This value is based on the number of M_w blocks that are sent to the cloud. The overall local space occupancy, Q, of all fog nodes can be expressed as a weighted sum based on their assigned weights β_{s_i} as shown in (13).

$$Q_{s_i} = \frac{e^{\left(\frac{N - M_W}{N}\right)}}{e - 1}. \tag{12}$$

$$Q = \sum_{i=1}^{m} Q_{s_i} \times \beta_{s_i}. \tag{13}$$

3.3 Multi-objective formulation

Based on the objective functions expressed earlier in the query probability, the storage cost, and the local space occupancy, the block selection problem can be formulated as a minimization multi-objective problem. This work's primary goal is to minimize the objective functions while taking as many blocks as possible to be stored in the cloud. Thus, all objective functions would be minimized, as shown in (14)–(16).

$$\begin{aligned} \min\left(query\, probs.\right) \Rightarrow \min & \sum_{j=1}^{M_{s_1}} P'_{b_j} \\ \min & \sum_{j=1}^{M_{s_i}} P'_{b_j} \\ \min & \sum_{j=1}^{M_{sm}} P'_{b_j}. \end{aligned} \tag{14}$$

From (14), it can be seen that there will be m fog nodes and thus m objective functions for the query probabilities, i.e., one for each fog node. The objective functions for the storage cost and the local space occupancy also minimized, as shown in (15) and (16).

$$\min kC \times \sum_{w=1}^{m} M_{\mathrm{w}}. \tag{15}$$

$$\min \sum_{i=1}^{m} Q_{s_i} \times \beta_{s_i}. \tag{16}$$

$$1 \le M_w \le N, \quad M_w \in \mathbb{N}. \tag{17}$$

Thus, for every set of m fog nodes that we have, it can be noted there would always be $m+2$ objective functions that we have to adhere to at every time. Users and operators of the Blockchain-IoT applications can always have constraints on these objective functions and the individual variables used in them. The constraints can be represented with $\gamma_1, \gamma_2, \ldots, \gamma_{m+2}$ as shown in (18). The constraints are solely the decision of the operator or the user. It must also be noted that the number of blocks that can be taken to the cloud must

$$\begin{aligned} &\max\left(\min \sum_{j=1}^{M_{s_1}} P'_{b_j}\right) \le \gamma_1 \\ &\max\left(\min \sum_{j=1}^{M_{sm}} P'_{b_j}\right) \le \gamma_m \\ &\max\left(\min kC \times \sum_{w=1}^{m} M_w\right) \le \gamma_{m+1} \\ &\vdots \\ &\max\left(\min \sum_{i=1}^{m} Q_{s_i} \times \beta_{s_i}\right) \le \gamma_{m+2} \\ &\text{s.t.} m. \end{aligned} \tag{18}$$

always be an integer and not a decimal number. Thus, M_{w} cannot be less than 1 as shown in (17).

The objective functions and multi-objective problem formulation that has been outlined in this section would be solved using the proposed advanced multi-objective particle swarm optimization approach, outlined in Sect. 4.

4 Methods

4.1 Particle swarm optimization (PSO) algorithm

In general, the particle swarm optimization algorithm is a widespread evolutionary algorithm that has been used a lot to solve single-optimization problems [42]. This algorithm is a random search method based on the behavioral pattern that simulates the way a swarm of birds forage and flock together. It has aspects that influence it based on the individual and social behavior of the birds (i.e., particles) in the swarm.

The PSO algorithm works in stages that include: initialization, searching, and updating best positions and values, converging with the best search results. All these processes are done over several iterations. The initialization phase of PSO is when a random set of particles or solutions to an optimization problem is generated. That is supposed to be representative of a swarm of birds. In the block selection problem, the initialization stage would give the random solutions for the number of blocks represented by H_f that should be sent to the cloud. The selection is based on the constraint of the number of blocks N each peer can hold as shown in (19) and a given population side, *Pop*, which the user provides.

$$H_\mathrm{f} = \begin{pmatrix} M_{s_1}^1 & M_{s_2}^1 & \cdots & M_{s_m}^1 \\ \vdots & & \ddots & \vdots \\ M_{s_1}^z & M_{s_2}^z & \cdots & M_{s_m}^z \end{pmatrix}. \quad (1 \le z \le Pop), \left(1 \le M_{s_i}^z \le N\right), M_{s_i}^z \in \mathbb{N}. \tag{19}$$

The objective functions are also evaluated at this stage using the generated particles, and the results obtained from the objective functions are used to judge the fitness of the solutions. The next stage of searching and updating the solutions or particles' positions and values occurs by altering the search direction of the solutions to move them to the best solution in the search space. These alterations and adjustments are made based on the two extremums, the personal best solution (P_k^i) of each particle i and the global best solution of the whole swarm (P_k^g). This is done in such a way that each particle in the swarm adjusts its position and direction to move toward the best result or solution that has been found and recorded by the whole swarm, which ensures that at the end of K iterations, all solutions would be converging toward the best solution. The adjustments made to the velocity, v_k^i, and position, x_k^i of each particle i during every kth iteration are shown in (20) and (21).

$$v_{k+1}^i = \omega \times v_k^i + c_1 r_1 \left(P_k^i - x_k^i\right) + c_2 r_2 \left(P_k^g - x_k^i\right). \tag{20}$$

$$\begin{aligned} & x_{k+1}^i = x_k^i + v_{k+1}^i \\ & 1 \le k \le K \\ & 1 \le i \le Pop. \end{aligned} \tag{21}$$

From (20) and (21), ω represents the inertia weight which works with velocity to improve the calculation speed and the quality, i.e., makes it converge faster or slower. r_1

and r_2 represent random values that help incorporate and model the aspect of environmental interference that birds in a swarm may face $(0 \leq r_1, r_2 \leq 1)$. c_1 and c_2 are known as the acceleration coefficients or learning factors [43]. Each iteration produces a variation in P_k^i, which is the personal best solution of a particle until the maximum number of iterations is reached.

The significant advantage of the PSO algorithm is its fast convergence [42], but the major setback is that it is notoriously known to get trapped into the local optima, which sometimes ends up in excessive searching.

4.2 Time-variant inertia weight PSO with variable-period cosine function

Several methods have been proposed to improve the already impressive convergence of PSO and prevent excessive searching and trapping in the local optima. Some of these methods include the Random Inertia Weight (RIW) method [43], the linear time-variant inertia weight [44], and the nonlinear time-variant inertia weight combined with a quadratic function introduced by Chatterjee et al. [45].

Zhang et al. [46] proposed a time-variant inertia weight combined with a variable-period cosine function. Their proposal was a multistage inertia weight method that has three main stages. The initial stage where the inertia is kept large for the global search process prevents the PSO algorithm from getting trapped in the local optima. The intermediate stage where the inertia value drops, and there is a transition from the global search process to the local search process. A final stage where the inertia value is kept small till the maximum iterations are reached so that the PSO can converge to an accurate value. The mathematical formulation for the multistage inertia weight $\omega_{\cos}(t)$, is shown (22).

$$\begin{aligned}
&\omega_{\cos}(k) = \frac{(\omega_{\text{ini}} + \omega_{\text{fin}})}{2} + \frac{(\omega_{\text{ini}} - \omega_{\text{fin}})}{2} \times \cos\left(\frac{I(k)\pi}{K}\right). \\
&I(k+1) = I(k) + a, I(1) = 0. \\
&a = \begin{cases} a_1, & \frac{I(k) \leq K}{6} \\ a_2, & \frac{K}{6} \leq I(k) \leq \frac{5K}{6} \\ a_3, & \frac{K}{6} < I(k) \leq K \end{cases}.
\end{aligned} \tag{22}$$

A term $I(k)$ is introduced into the cosine function to adjust the period, and it changes on every iteration. To keep track of the stage at which the PSO algorithm is in, another variable a is introduced to help update $I(k)$. Based on experiments performed in [46], the values chosen for the a_i values are as follows; $a_1 = \left(\frac{4}{3}\right), a_2 = \left(\frac{16}{3}\right)$ and $a_3 = \left(\frac{2}{9}\right)$. A user can specify the starting initial inertia weight ω_{ini} and final inertia weight ω_{fin}, but it is proposed to be set to $\omega_{\text{ini}} = 0.4$ and $\omega_{\text{fin}} = 0.9$.

4.3 Multi-objective particle swarm optimization (MOPSO)

MOPSO was first introduced by Coello and Lechuga [38] to deal with multi-objective optimization problems. Before its introduction, PSO could only handle single-objective optimization problems. Multi-objective optimization problems had to be condensed into single-objective problems using the weighted sum approach [47].

Two particles can determine which particle is the best for single-objective problems, but that is not the case when dealing with multi-objective problems. This is because there is more than one objective function and thus multiple fitness values to compare.

Thus, the concept of the Pareto-optimality and dominance is introduced. This helps to tell whether a particle is better than another by ensuring each particle's targets are better than the current best, not one or two but all.

In MOPSO, apart from the updated population during every iteration, an external archive keeps track of non-dominated solutions (i.e., solutions with the number blocks to be sent to the cloud for each peer). These archive elements are referred to as the Pareto set, and the eventual goal is to have a Pareto optimal set. Elements are added to this archive until it is complete, and then the archive is sorted, and only the very best solutions are kept. This is where the update would be made to MOPSO for this paper.

When it comes to the global best search result, there would be more than one optimal solution because there are multiple objective functions described in Sect. 3. Thus, a global leader is chosen and used to adjust all other solutions. The flow chart for the MOPSO algorithm is shown in Fig. 4.

4.4 Advanced time-variant multi-objective particle swarm optimization (AT-MOPSO) algorithm

Based on research on time-variant inertia weight and non-dominated Pareto sets, AT-MOPSO is proposed to improve the original MOPSO.

AT-MOPSO makes use of the time-variant inertia weight suggested in [46] to improve in the update process for the position (x^i_{k+1}) and velocity (v^i_{k+1}) values by the incorporating $\omega_{\cos}(k)$ as a replacement to the fixed ω in the original MOPSO equation.

The next update to the MOPSO is the incorporation of non-dominated sorting and mutation of the external archive $H_{\mathrm{f}}^{(k)}$ adapted from the Non-Dominated Sorting Genetic Algorithm [48]. The mutation process randomly takes a subset of $H_{\mathrm{f}}^{(k)}$, usually $Pop/2$, i.e., 50% of the total number of the external archive. For each solution, a set number of particles are picked at random. The position values for those particles are mutated using a mutation rate (μ) and mutation step size (σ), as shown in Fig. 5.

The updated positions are then based on the objective functions where the values are calculated and the number of blocks per fog node. The algorithm for the mutation is given in Algorithm 1.

After the mutant population (popm) is created, a joint set of the mutant population (popm) and the external archive $H_{\mathrm{f}}^{(k)}$ is created, and this set is sorted, and the non-dominated solutions are found. The external archive $H_{\mathrm{f}}^{(k)}$ is then updated with these non-dominated solutions. The entire algorithm for AT-MOPSO is given in the flowchart in Fig. 6. The major areas which have been improved and added to the original algorithm are indicated in orange.

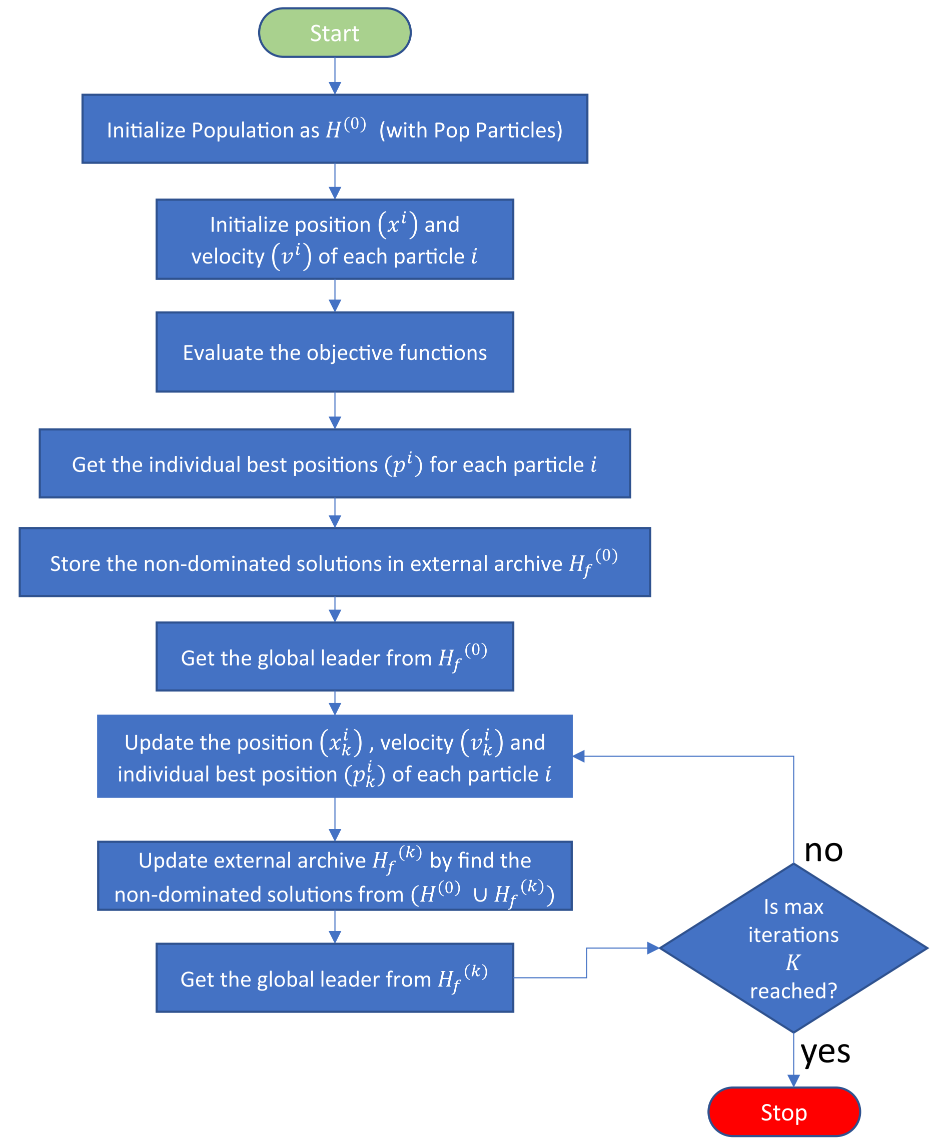
Start
Initialize Population as $H^{(0)}$ (with Pop Particles)
Initialize position (x^i) and velocity (v^i) of each particle i
Evaluate the objective functions
Get the individual best positions (p^i) for each particle i
Store the non-dominated solutions in external archive $H_f^{(0)}$
Get the global leader from $H_f^{(0)}$
Update the position (x_k^i), velocity (v_k^i) and individual best position (p_k^i) of each particle i
Update external archive $H_f^{(k)}$ by find the non-dominated solutions from $(H^{(0)} \cup H_f^{(k)})$
Get the global leader from $H_f^{(k)}$
Is max iterations K reached?
no
yes
Stop

Fig. 4 Flow chart for the original MOPSO algorithm

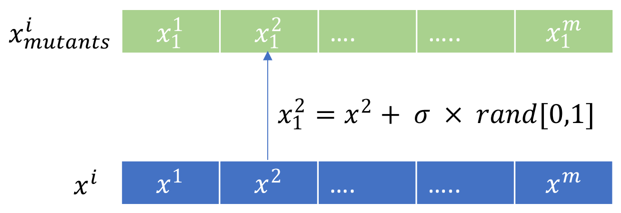

Fig. 5 Mutation operation to create mutant particles

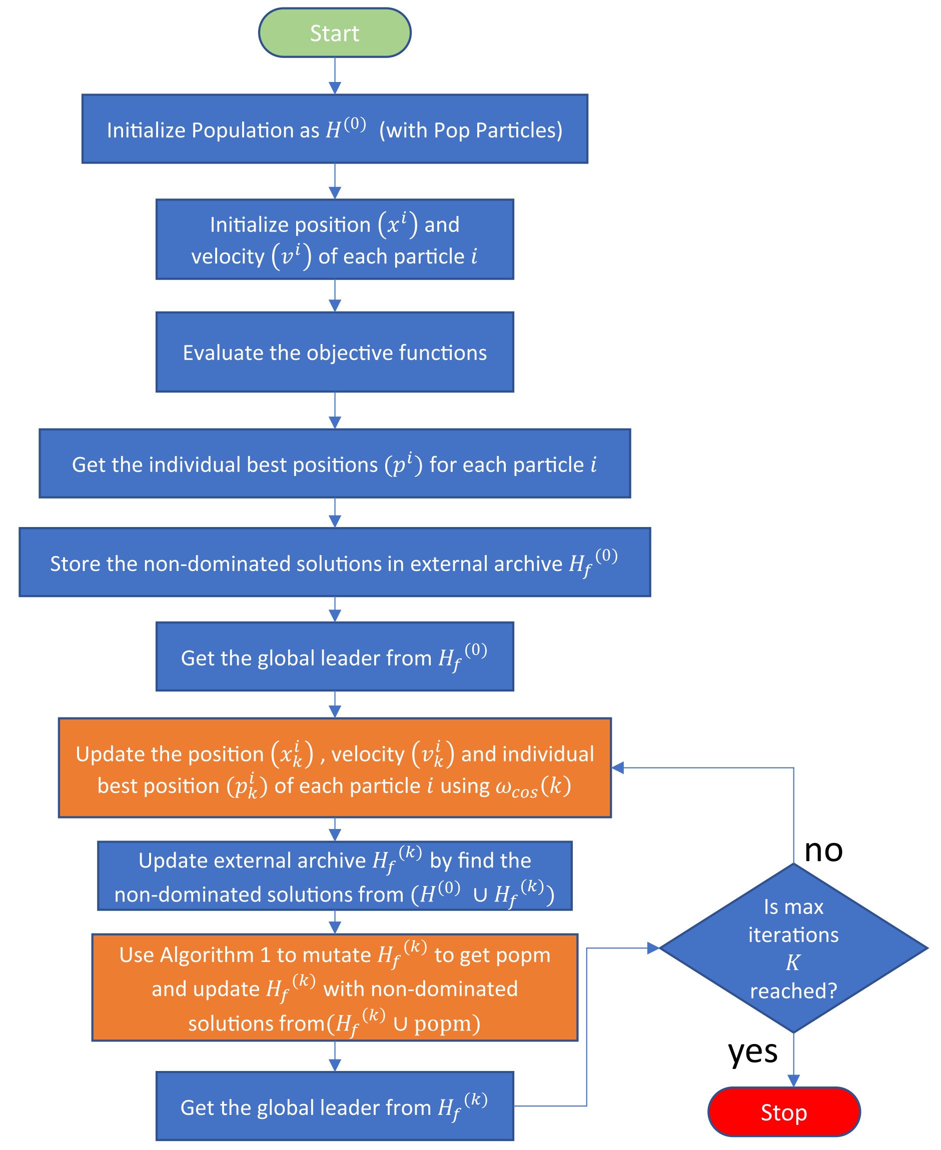
Start
Initialize Population as $H^{(0)}$ (with Pop Particles)
Initialize position (x^i) and velocity (v^i) of each particle i
Evaluate the objective functions
Get the individual best positions (p^i) for each particle i
Store the non-dominated solutions in external archive $H_f{}^{(0)}$
Get the global leader from $H_f{}^{(0)}$
Update the position (x_k^i), velocity (v_k^i) and individual best position (p_k^i) of each particle i using $\omega_{cos}(k)$
Update external archive $H_f{}^{(k)}$ by find the non-dominated solutions from $(H^{(0)} \cup H_f{}^{(k)})$
Use Algorithm 1 to mutate $H_f{}^{(k)}$ to get popm and update $H_f{}^{(k)}$ with non-dominated solutions from$(H_f{}^{(k)} \cup \text{popm})$
Get the global leader from $H_f{}^{(k)}$
Is max iterations K reached?
no
yes
Stop

Fig. 6 Flow chart for AT-MOPSO which is a proposed update to MOPSO

Algorithm 1 – Mutation Algorithm for AT-MOPSO

```
Let popm = population for the mutants to be created
Size(popm) = $^{Pop}/_2$

for y = 1:size(popm)
        i = random integer from [1 Size($H_f^{(k)}$ )]
        x = $H_f^{(k)}$[i] /* one solution is selected from external archive */

        nVar = numel (p) /* number of elements from selected solution = number of fog nodes available */

        nMu = $\mu$ x nVar /*number of particles to mutate*/
        j = randsample (nVar, nMu) /*randomly draw indices for nMu samples */

        x_mutant = x
        x_mutant [j] = x [j] + $\sigma$ $\times$ rand(size(j)) /* mutate values at selected indices in x_mutant */
        popm[y] = x_mutant
end
```

It must be noted that actual values will be put in the population $H^{(0)}$, the external archive $H_{\mathrm{f}}^{(k)}$. The mutant population (popm) would all have to be integers since these solutions represent the number of blocks M_{s_i} for each fog node that would be sent to the cloud. Thus, an extra step of ensuring that all solutions are converted to integers must be guaranteed across the whole algorithm for this use case.

5 Experiments and results

The AT-MOPSO algorithm is applied to the objective functions outlined in Sect. 3 of this paper, shown in (14)–(18). When the maximum number of iterations is reached, the external archive H_f which contains the final Pareto set of non-dominated solutions, is filtered using the constraints set for each objective function $\gamma_1, \gamma_2, \ldots, \gamma_{m+2}$. If any of the solutions has a corresponding objective function value less than γ_i, it is filtered out and removed from the external archive into a new set H_f^*. This would allow only for solutions that satisfy all constraints to be in H_f.

After the filtration step, we would still be left with some, Pareto optimal solutions, but we need the best solution out of the set H_f^* to answer our research questions of how many blocks for each fog node must be sent to the cloud. This was done by iteratively going through each solution in H_f^* and finding the minimum weighted sum of all objective functions using (23) where $\delta_l (1 \le l \le m+2)$ represents the weights assigned to the individual objective function.

$$\mathrm{WS} = \sum_{j=1}^{m} \delta_j P'_{b_j} + \delta_{m+1} \cos t + \delta_{m+2} Q. \tag{23}$$

To show our AT-MOPSO algorithm's potency, we ran it. We compared the results for the different objective functions against the original MOPSO, time-variant MOPSO (offered as I-MOPSO), and NSGA-III (shown as NSGA3). The experiments

Table 1 Computer specifications that was used for the experiments

Computer component	Specification value
Processor	Intel Xeon E5-2690
Processor core count	4
Processor clock frequency	2.9 GHz
RAM	32 GB
Workers in parallel pool (MATLAB)	12

Table 2 List of parameters used for experiment for the different algorithms

Parameter	Value
F_0	0.95
N	200
β_{s_1}	0.3
β_{s_2}	0.2
β_{s_3}	0.5
γ_1	0.45
γ_2	0.45
γ_3	0.45
γ_4	3.0
γ_5	0.6
δ_1	0.16
δ_2	0.16
δ_3	0.16
δ_4	0.12
δ_5	0.4

used for testing the algorithms were done on a computer with an Intel Xeon E5 processor with 2.9 GHz and 32 GB of RAM (the full specifications of the computer is shown in Table 1).

To simplify the experiment, we assumed a fixed query frequency F_0 of 0.95. The set value helps to depict a typical use case involving IoT fog nodes connected to a Blockchain with IoT devices being used in a traceability application such as [33] as described in Sect 3. Three fog node servers were selected; thus, the set $S = \{s_1, s_2, s_3\}$ and the number of blocks to be sent to the cloud for each fog node is M_{s_1}, M_{s_2}, and M_{s_3} respectively, making $m = 3$. For simplicity, the total number of blocks N that can be stored for each fog node was taken to be 200 at a 1 MB size for each block. The local space occupancy weights for the individual fog nodes were set as β_{s_1}, β_{s_2}, and β_{s_3}. The constraints for the five objective functions (i.e., $m + 2$) were specified as $\gamma_1, \gamma_2, \gamma_3, \gamma_4$, and γ_5. Also, the weights $\delta_1, \delta_2, \delta_3, \delta_4$ and δ_5 of each objective function for the weighted sum computation in (23) was also specified. A complete list of all parameters used for the experiment is specified in Table 2.

For the implementations, the total number of iterations K was set to 200. For each iteration, averages of the solutions in the external archive were computed, and graphs plotted for *F1, F2, F3, F4,* and *F5*, representing each objective function as shown in Fig. 7. It must be noted that it takes some time for the algorithms to run for the given problem.

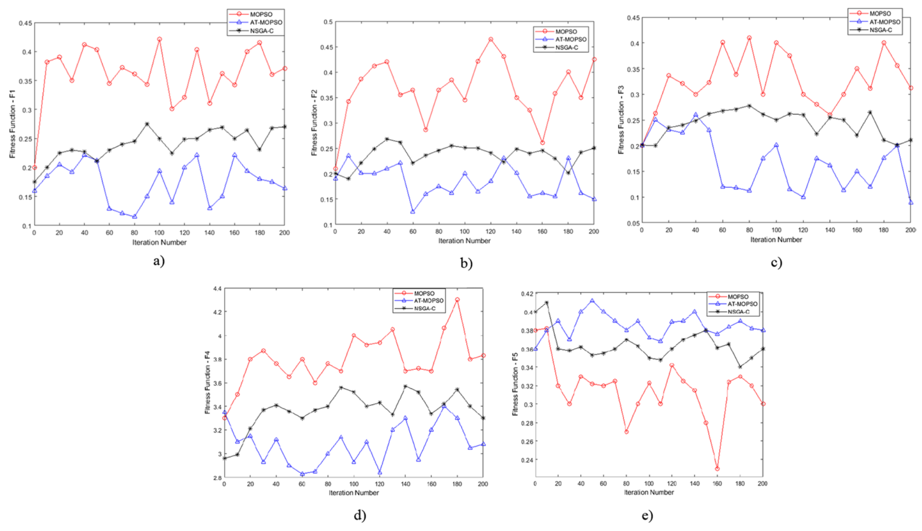

Fig. 7 The graphs showing the performance of the objective functions for the different objective functions (i.e., F1, F2, F3, F4 and F5) when AT-MOPSO was used

This problem is not being run to the point of full convergence but rather to find the algorithm that can minimize the objective functions as much as possible in the run time.

The graph in Fig. 7 shows the results for the when the AT-MOPSO was compared to the original MOPSO algorithms as well as the NSGA-C (Non-Dominated Sorting Genetic Algorithm with Clustering). There is not research existing the direction of using optimization techniques to find solve this cloud storage block selection problem. The most recent scheme that has been used to solve this is the NSGA-C based on the NSGA-III algorithm proposed by Xu et al. [21]. It can be clearly shown from the results that our proposed scheme (AT-MOPSO) out-performed NSGA-C and the original MOPSO algorithm for the first four objective functions (i.e., *F1–F4*). It can also be observed that for the objective function *F5*, AT-MOPSO performs slightly worse than NSGA-C and the original MOPSO. However, the advantage AT-MOPSO and other PSO algorithms have over NSGA-III is they converge at a faster rate.

Thus, the results of the AT-MOPSO being slightly worse in F5 in comparison with NSGA-C, still gives it an edge over the NSGA-C due to the afore mentioned reasons. The benefits of the AT-MOPSO can be further elaborated by considering the energy efficiency or energy-saving analysis of the two algorithms.

We performed further investigations into how our proposed scheme compares to some of the common and popular Multi-Objective algorithms. In Fig. 8, we show our results when our developed scheme was compared to SPEA-II (Strength Pareto Evolutionary Algorithm) [49], PESA-II (Pareto Envelope-based Selection Algorithm with region-based selection) [50] and NSGA-III. Our scheme also out performs the SPEA-II and PESA-II algorithms with SPEA-II performing slightly better than PESA-II for the same block selection problem using the same parameters.

It can also be seen in Fig. 8, that our developed scheme (AT-MOPSO) is at par with NSGA-III in most of the objective functions (i.e., *F1–F4*) and even marginally out-performing NSGA-III in objective function *F4*. AT-MOPSO is also out-performed by

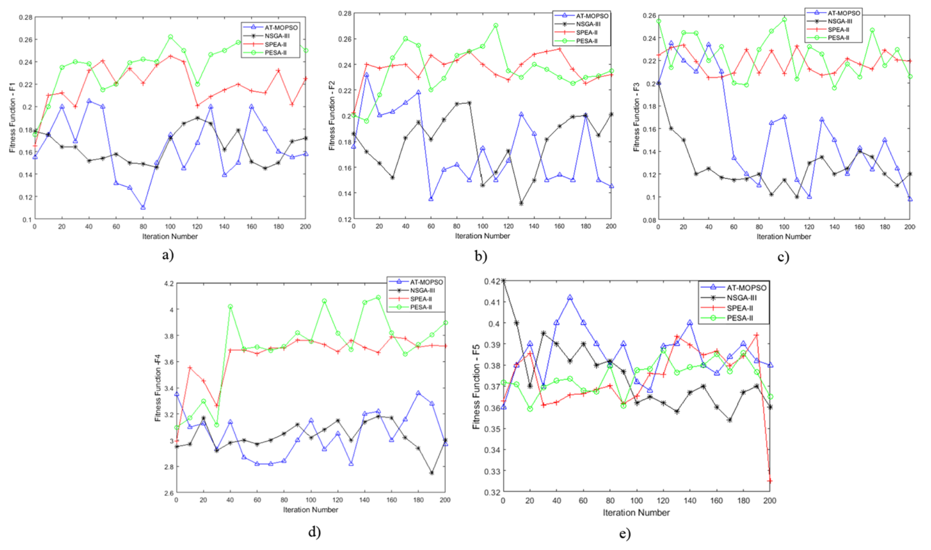

Fig. 8 Graphs showing how the AT-MOPSO algorithm performs in comparison to other well-known multi-objective optimization scheme

NSGA-III in the results for objective function *F5*. For the same afore mentioned as NSGA-C (which is based on NSGA-III), AT-MOPSO should be selected due to its faster run time and convergence rate.

6 Discussion

We discuss the potency of our develop AT-MOPSO algorithm by taken a good look a critical energy efficiency comparison with other Multi-Objective Optimization algorithms. All the specifications of the computing resources used for this comparison is shown in Table 1. Shukla et al. [46] used the power consumption model for CMOS (Complementary Metal Oxide Semiconductor) logic circuits to analyze dynamic energy consumption on microprocessors. They considered that for microprocessors, the capacitive power or the dynamic energy consumption is the most significant factor, and thus they expressed it mathematically as shown in (24):

$$\begin{aligned} E_{\text{busy}} &= \sum_{i=1}^{n} B \times v_i^2 \times f_r \times t_i \\ E_{\text{idle}} &= \sum_{i=1}^{n} B \times v_{i_{\text{lowest}}}^2 \times f_{r_{\text{lowest}}} \times t_{i_{\text{idle}}} \\ E_{\text{total}} &= E_{\text{busy}} + E_{\text{idle}} \end{aligned} \tag{24}$$

where B is the constant parameter related to the dynamic power (based on DVFS—Dynamic Voltage and Frequency Scaling) for a given CPU, v_i is the supply voltage at which the processor is regulated, f_r is the clock frequency of the microprocessor, and t_i is the time for which a microprocessor runs a task.

Table 3 Results for the runtimes in seconds for different algorithms compared

Optimization algorithm	Runtime (s)
MOPSO	371.8
NSGA-C	872.4
AT—MOPSO	384.2
NSGA3	758.8

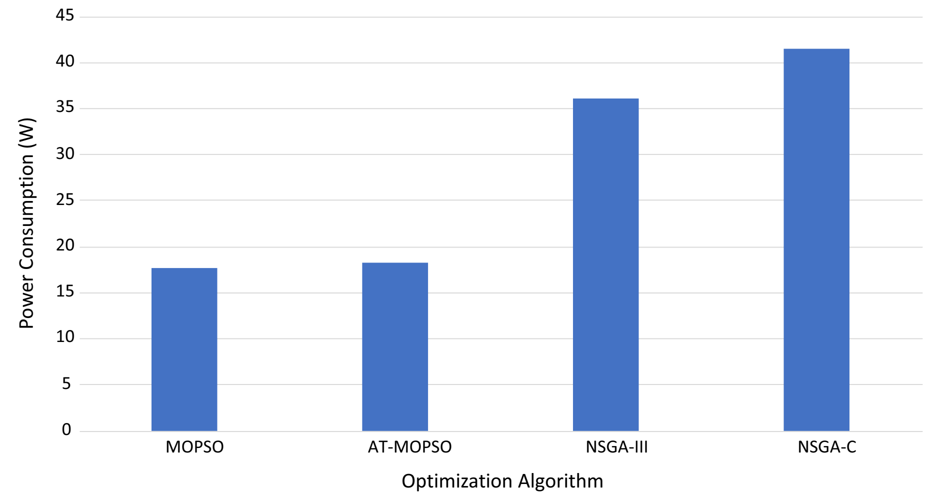

Fig. 9 Results of energy consumption of the different optimization algorithms

There were two considered cases where the processor is in an idle state E_{idle} and when the processor is running a task E_{busy}. The times for which the various algorithms run the 200 iterations were recorded and are shown in Table 3. It can be seen from the results that the AT-MOPSO runs in also half the time it takes the NSGA-III algorithm to complete the 200 iterations with the given parameters. The time it takes to complete the same number of iterations using NSGA-C (the most recent scheme used to solve the block selection problem) is even more than that of NSGA-III. Xu et al. [21] considered their scheme to be a better approach due to the fact that it outperformed NSGA-III for the F5 objective function which represents that for local space occupancy (arguably the important objective).

We have succeeded in showing that our scheme out-performs NSGA-C in all other objective functions and further performs in an almost similar fashion as NSGA-C for objective function F5, yet running in less than half the time it takes to run NSGA-C. The recorded execution times were fed into (24) to calculate the energy consumed the runtime of each algorithm. The results for the energy consumed executing each of the algorithms are shown in the graph in Fig. 9. The algorithms' energy consumption is an essential factor to consider because this is a task the processor of the fog node would be running on a persistent basis, and having an optimization algorithm that can converge very fast and use less energy is an excellent feature to have.

7 Application of the at-MOPSO to a real-world industrial blockchain system for maintenance

To show the true potency of the developed algorithm and block selection approach, we incorporated it into a real-world industrial Blockchain maintenance system which was shown, proposed and tested in the BISS 4.0 platform [23]. We adapted the architecture and showed where the AT-MOPSO algorithm could prove useful and is shown in Fig. 10

In the original implementation of the BISS 4.0 platform, there is a combination of enterprise architectures, services and data to build a Blockchain system. The implemented Blockchain in the architecture is Hyperledger Fabric [51] which is a permissioned Blockchain, thus all the peers in the architecture can know each other without any form of trust since a permissioned Blockchains ensures the user groups that have access to the Blockchain is restricted.

The BISS platform has sensors and actuators being joined and connected to machines. These machines are in turn connected through an adapter (running a fabric client) to the main computing device that acts as a peer on the Blockchain. This is done so that all storage and computing tasks can be delegated to that peer. The downside of such an architecture is that the machine is not truly (but virtually) connected to the Blockchain. This approach of incorporating a Blockchain in an IoT enterprise system is one that is very common in industrial IoT systems. The said approach is very common due to the storage demands and processing power needed to run the machines that the sensors and actuators are connected to, due to the less potent nature of such machines. This is clearly shown in the BISS platform architecture.

By the use of the approach introduced in this paper, those machines can be brought closer to the Blockchain such that each machine can then act as a peer on the Blockchain. This means that machine with cheaper hardware and minimal storage space can be used as peers on the Blockchain. Our approach would enable this to happen because there would be a lower storage demand on the machine such that some of their blocks would be moved to the cloud for storage as shown in Fig. 10.

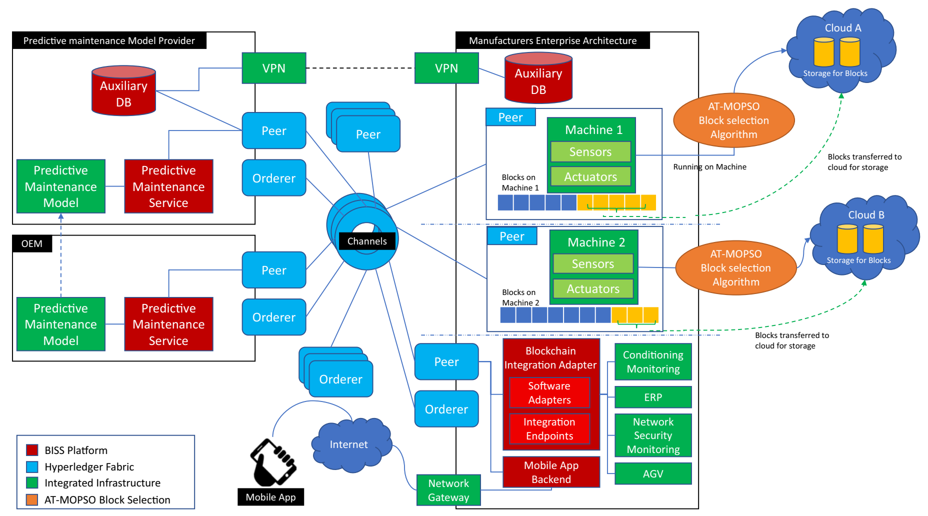
Predictive maintenance Model Provider
Auxiliary DB
VPN
Peer
Orderer
Predictive Maintenance Model
Predictive Maintenance Service
OEM
Peer
Orderer
Predictive Maintenance Model
Predictive Maintenance Service
Peer
Channels
Orderer
Manufacturers Enterprise Architecture
VPN
Auxiliary DB
Peer
Machine 1
Sensors
Actuators
Blocks on Machine 1
Peer
Machine 2
Sensors
Actuators
Blocks on Machine 2
Peer
Orderer
Blockchain Integration Adapter
Software Adapters
Integration Endpoints
Mobile App Backend
Conditioning Monitoring
ERP
Network Security Monitoring
AGV
Network Gateway
Internet
Mobile App
Cloud A
Storage for Blocks
AT-MOPSO Block selection Algorithm
Running on Machine
Blocks transferred to cloud for storage
Cloud B
Storage for Blocks
AT-MOPSO Block selection Algorithm
Blocks transferred to cloud for storage
🟥 BISS Platform
🟦 Hyperledger Fabric
🟩 Integrated Infrastructure
🟧 AT-MOPSO Block Selection

Fig. 10 The improved BISS platform architecture with the AT-MOPSO incorporated and running on peer

It must also be noted that in the incorporation of Blockchain into industrial systems and machines as shown in the BISS architecture, the machines are mainly going to be legacy and older systems so having such an algorithm to take some of the pressure of them would be greatly appreciated by operators of such systems and provide a true connection between the machines and the Blockchain.

In the version of BISS which incorporates our approach, machines which house the sensors and actuators are now peers on the Blockchain and they would in turn be running the AT-MOPSO algorithms. From our pilot testing of the algorithm, it was observed that as the machines run the given algorithm, they are able to offload some of the blocks that are produced from transactions in their ledger to the cloud and this reduced the transaction time by 52%. The is because, the AT-MOPSO takes into consideration (as a parameter), the amount of storage space that the peer on the Blockchain has available, thus the optimization is done based on the provided value. The three main objective functions that the block selection is based on takes into account the cost of storage of the blocks in the cloud as well as local storage occupancy. All these aforementioned factors, make the developed and proposed algorithm one that can be put to great and diverse use as shown with this industrial application example.

8 Conclusion

In this paper, we looked at the integration of Blockchain with IoT applications. We proposed a hybrid Blockchain-IoT integration scheme that used fog computing and cloud storage to help improve the throughput of such applications. The scheme scheduled a side-chain run on a fog node and only sent completed transactions in the side-chain to the main Blockchain. The proposed scheme alleviates the storage pressure on fog nodes by ensuring that some of the blocks produced by the transactions are stored in the cloud for each connected fog node.

To select the number of blocks that should be sent to the cloud, we further proposed an Advanced Time-Variant Multi-Objective Particle Swarm Optimization (AT-MOPSO) algorithm to help solve the block selection problem. The algorithm was applied to objective functions formulated to model the aspects of the query probability of the blocks on each fog node, the cloud storage cost to be incurred by a user, and the local space occupancy/storage availability needed to be saved for each fog node. We compared our proposed algorithm to the original MOPSO algorithm and the Time-Variant MOPSO and NSGA-III. We observed that our scheme performed better in all objective functions than all other MOPSO algorithms except for the local space occupancy. Our AT-MOPSO algorithm also performed as well as the NSGA-III algorithm for the query probability objective function for the first and third fog nodes, and also for the cloud storage cost objective function. We also assessed our proposed AT-MOPSO algorithm to determine its energy-saving efficiency compared to the NSGA-III algorithm (which is the current standard or algorithm which has been used to solve this problem). Our algorithm runs in about half the time of the NSGA-III and achieved about 52% energy efficiency compared to the NSGA-III.

We further showed how our proposed algorithm can be incorporated into industrial systems which are running legacy machinery. This was shown but adapting the BISS platform architecture and showing where our AT-MOPSO would fit in it.

In our future work, we plan to explore the possibilities of reducing the algorithm's runtime time even further, thus allowing us to deal with fog nodes with larger storage sizes of about 1TB or more. It will also be worth looking at the effects of our algorithm on aspects of the IoT devices such as Quality of Service (QoS) and transmit power limitations.

Abbreviations
IoT: Internet of Things; DAG: Directed acyclic graph; PSO: Particle swarm optimization; MOPSO: Multi-objective particle swarm optimization; AT-MOPSO: Advanced time-variant multi-objective particle swarm optimization; NSGA: Non-dominated sorting genetic algorithm with clustering; MOEA: Multi-objective evolutionary algorithms; PV: Photo-voltaic.

Acknowledgements
The authors are grateful to the TWAS-DFG Visiting Researcher Programme for providing and funding a collaborative environment between Offenburg University of Applied Sciences and KNUST to undertake this research. Authors are also grateful to the KNUST-MTN Innovation Fund for providing support to undertake the research and pay the APC.

Authors' contributions
CN and ET conceived and designed the study. CN, ET, HN, JG and BY performed the experiments and wrote the paper. ET, DW and BY revised the manuscript. ET and CN took charge of all the work of paper submission. HN, JG, AS and DW gave several proposals for the experiments and interpretation of the results. JG, AS and BY reviewed and revised the manuscript. All authors read and approved the final manuscript.

Funding
This study received no external funding.

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated during the current study. All data generated or analyzed during this study are included in this article.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Engineering and IT, Carinthia University of Applied Sciences, Carinthia, Austria. [2]Department of Computer Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana. [3]Department of Telecommunications Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana. [4]The Institute of Reliable Embedded Systems and Communication Electronics, Offenburg University of Applied Sciences, Offenburg, Germany.

Received: 22 April 2021 Accepted: 26 November 2021
Published online: 04 January 2022

References
1. J. Xu et al., Healthchain: a blockchain-based privacy preserving scheme for large-scale health data. IEEE Internet Things J. **6**(5), 8770–8781 (2019)
2. P.P. Ray, D. Dash, K. Salah, N. Kumar, Blockchain for IoT-based healthcare: background, consensus, platforms, and use cases. IEEE Syst. J. **15**, 85–94 (2020)
3. S. Aich, S. Chakraborty, M. Sain, H. Lee, H.-C. Kim, A review on benefits of IoT integrated blockchain based supply chain management implementations across different sectors with case study, in *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pp. 138–141 (2019). https://doi.org/10.23919/ICACT.2019.8701910
4. G. Caldarelli, J. Ellul, Trusted academic transcripts on the blockchain: a systematic literature review. Appl. Sci. (2021). https://doi.org/10.3390/app11041842
5. J.H. Huh, S.K. Kim, Verification plan using neural algorithm blockchain smart contract for secure P2P real estate transactions. Electronics (2020). https://doi.org/10.3390/electronics9061052
6. L. Cui et al., A blockchain-based containerized edge computing platform for the internet of vehicles. IEEE Internet Things J. **8**(4), 2395–2408 (2021). https://doi.org/10.1109/JIOT.2020.3027700
7. S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system. *Bitcoin*. https://bitcoin.org/bitcoin.pdf
8. I. Eyal, A.E. Gencer, E.G. Sirer, R.V. Renesse, Bitcoin-NG: a scalable blockchain protocol, in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, Santa Clara, CA, pp. 45–59 (2016). https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal
9. A. Carvalho, J.W. Merhout, Y. Kadiyala, J. Bentley II., When good blocks go bad: managing unwanted blockchain data. Int. J. Inf. Manag. **57**, 102263 (2021). https://doi.org/10.1016/j.ijinfomgt.2020.102263
10. Bitcoin blockchain size 2009–2021. Statista. https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/. Accessed 24 Mar 2021

11. V. Buterin, Ethereum whitepaper. *Ethereum*. https://ethereum.org/whitepaper/
12. S. Li, M. Yu, C. Yang, A.S. Avestimehr, S. Kannan, P. Viswanath, PolyShard: coded sharding achieves linearly scaling efficiency and security simultaneously, Cornell University (2020)
13. J. Poon, V. Buterin, Plasma: scalable autonomous smart contracts (2017). http://plasma.io/
14. U. Klarman, S. Basu, A. Kuzmanovic, E.G. Sirer, bloXroute: a scalable trustless blockchain distribution network," *BloXroute Labs* (2019). https://bloxroute.com/wp-content/uploads/2019/11/bloXrouteWhitepaper.pdf
15. C. Jin, S. Pang, X. Qi, Z. Zhang, A. Zhou, A high performance concurrency protocol for smart contracts of permissioned blockchain. IEEE Trans. Knowl. Data Eng. (2021). https://doi.org/10.1109/TKDE.2021.3059959
16. S. Mercan, A. Kurt, E. Erdin, K. Akkaya, Cryptocurrency solutions to enable micro-payments in consumer IoT. IEEE Consum. Electron. Mag. (2021). https://doi.org/10.1109/MCE.2021.3060720
17. M. Abbasi, E. Mohammadi-Pasand, M.R. Khosravi, Intelligent workload allocation in IoT–Fog–cloud architecture towards mobile edge computing. Comput. Commun. **169**, 71–80 (2021). https://doi.org/10.1016/j.comcom.2021.01.022
18. Y. Zhang et al., High-performance isolation computing technology for smart IoT healthcare in cloud environments. IEEE Internet Things J. (2021). https://doi.org/10.1109/JIOT.2021.3051742
19. J. Zhang, X. Zhou, T. Ge, X. Wang, T. Hwang, Joint task scheduling and containerizing for efficient edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(8), 2086–2100 (2021). https://doi.org/10.1109/TPDS.2021.3059447
20. R. Yang, F.R. Yu, P. Si, Z. Yang, Y. Zhang, Integrated blockchain and edge computing systems: a survey, some research issues and challenges. IEEE Commun. Surv. Tutor. **21**(2), 1508–1532 (2019). https://doi.org/10.1109/COMST.2019.2894727
21. M. Xu, G. Feng, Y. Ren, X. Zhang, On cloud storage optimization of blockchain with a clustering-based genetic algorithm. IEEE Internet Things J. **7**(9), 8547–8558 (2020). https://doi.org/10.1109/JIOT.2020.2993030
22. C. Nartey et al., On blockchain and IoT integration platforms: current implementation challenges and future perspectives. Wirel. Commun. Mob. Comput. **2021**, e6672482 (2021). https://doi.org/10.1155/2021/6672482
23. D. Welte, A. Sikora, D. Schönle, J. Stodt, C. Reich, Blockchain at the shop floor for maintenance, in *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 15–22 (2020). https://doi.org/10.1109/CyberC49757.2020.00013
24. X. Cai et al., A sharding scheme based many-objective optimization algorithm for enhancing security in blockchain-enabled industrial Internet of Things. IEEE Trans. Ind. Inform. (2021). https://doi.org/10.1109/TII.2021.3051607
25. V. Buterin, A next-generation smart contract and decentralized application platform, ethereum.org (2014). Accessed 24 Mar 2021
26. S. King, S. Nadal, PPCoin: peer-to-peer crypto-currency with proof-of-stake. *Chain Extranet*. https://www.chainwhy.com/upload/default/20180619/126a057fef926dc286accb372da46955.pdf
27. D. Schwartz, N. Youngs, A. Britto, The ripple protocol consensus algorithm. *Ripple*. https://ripple.com/files/ripple_consensus_whitepaper.pdf
28. I. Sfiligoi, D. Schultz, F. Würthwein, B. Riedel, Pushing the cloud limits in support of IceCube science. IEEE Internet Comput. **25**(1), 71–75 (2021). https://doi.org/10.1109/MIC.2020.3045209
29. Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, Q. Ni, Deploying edge computing nodes for large-scale IoT: a diversity aware approach. IEEE Internet Things J. **5**(5), 3606–3614 (2018). https://doi.org/10.1109/JIOT.2018.2823498
30. P. Lin, Q. Song, D. Wang, R. Yu, L. Guo, V. Leung, Resource management for pervasive edge computing-assisted wireless VR streaming in industrial Internet of Things. IEEE Trans. Ind. Inform. (2021). https://doi.org/10.1109/TII.2021.3061579
31. A. Palai, M. Vora, A. Shah, Empowering light nodes in blockchains with block summarization, in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5 (2018). https://doi.org/10.1109/NTMS.2018.8328735
32. S. Kadhe, J. Chung, K. Ramchandran, SeF: a secure fountain architecture for slashing storage costs in blockchains (2019). Accessed 24 Mar 2021. http://arxiv.org/abs/1906.12140
33. X. Yang, M. Li, H. Yu, M. Wang, D. Xu, C. Sun, A trusted blockchain-based traceability system for fruit and vegetable agricultural products. IEEE Access **9**, 36282–36293 (2021). https://doi.org/10.1109/ACCESS.2021.3062845
34. A. Kumari, R. Gupta, S. Tanwar, Amalgamation of blockchain and IoT for smart cities underlying 6G communication: a comprehensive review. Comput. Commun. **172**, 102–118 (2021). https://doi.org/10.1016/j.comcom.2021.03.005
35. Y. Hassanzadeh-Nazarabadi, N. Nayal, S.S. Hamdan, Ö. Özkasap, A. Küpçü, A containerized proof-of-concept implementation of LightChain system, in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–2 (2020). https://doi.org/10.1109/ICBC48266.2020.9169463
36. Y. Yang, J. Liu, S. Tan, A multi-objective evolutionary algorithm for steady-state constrained multi-objective optimization problems. Appl. Soft Comput. **101**, 107042 (2021). https://doi.org/10.1016/j.asoc.2020.107042
37. H.M. Ridha, C. Gomes, H. Hizam, M. Ahmadipour, A.A. Heidari, H. Chen, Multi-objective optimization and multi-criteria decision-making methods for optimal design of standalone photovoltaic system: a comprehensive review. Renew. Sustain. Energy Rev. **135**, 110202 (2021). https://doi.org/10.1016/j.rser.2020.110202
38. C.A.C. Coello, M.S. Lechuga, MOPSO: a proposal for multiple objective particle swarm optimization, in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2, pp. 1051–1056 (2002). https://doi.org/10.1109/CEC.2002.1004388
39. R. Xu, S.Y. Nikouei, Y. Chen, E. Blasch, A. Aved, BlendMAS: a blockchain-enabled decentralized microservices architecture for smart public safety, in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 564–571 (2019). https://doi.org/10.1109/Blockchain.2019.00082
40. K.P. Tsang, Z. Yang, The market for bitcoin transactions. J. Int. Financ. Mark. Inst. Money **71**, 101282 (2021). https://doi.org/10.1016/j.intfin.2021.101282
41. J. Sousa, A. Bessani, M. Vukolic, A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform, in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 51–58 (2018). https://doi.org/10.1109/DSN.2018.00018

42. Y. Wen-Bin, D. Yong-Hong, W-MOPSO in adaptive circuits for blast wave measurements. IEEE Sens. J. **21**(7), 9323–9332 (2021). https://doi.org/10.1109/JSEN.2021.3053099
43. Y.-H. Lin, L.-C. Huang, S.-Y. Chen, C.-M. Yu, The optimal route planning for inspection task of autonomous underwater vehicle composed of MOPSO-based dynamic routing algorithm in currents. Appl. Ocean Res. **75**, 178–192 (2018). https://doi.org/10.1016/j.apor.2018.03.016
44. M. Fan, M. Fan, Y. Akhter, A time-varying adaptive inertia weight based modified PSO algorithm for UAV path planning, in *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pp. 573–576 (2021). https://doi.org/10.1109/ICREST51555.2021.9331101
45. A. Chatterjee, P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Comput. Oper. Res. **33**(3), 859–871 (2006). https://doi.org/10.1016/j.cor.2004.08.012
46. J. Zhang, J. Sheng, J. Lu, L. Shen, UCPSO: a uniform initialized particle swarm optimization algorithm with cosine inertia weight. Comput. Intell. Neurosci. **2021**, e8819333 (2021). https://doi.org/10.1155/2021/8819333
47. S. Liang et al., Determining optimal parameter ranges of warm supply air for stratum ventilation using Pareto-based MOPSO and cluster analysis. J. Build. Eng. **37**, 102145 (2021). https://doi.org/10.1016/j.jobe.2021.102145
48. K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
49. R. Gharari, N. Poursalehi, M. Abbasi, M. Aghaie, Implementation of strength Pareto evolutionary algorithm II in the multiobjective burnable poison placement optimization of KWU pressurized water reactor. Nucl. Eng. Technol. **48**(5), 1126–1139 (2016). https://doi.org/10.1016/j.net.2016.04.004
50. D.W. Corne, N.R. Jerram, J.D. Knowles, M.J. Oates, PESA-II: region-based selection in evolutionary multiobjective optimization, in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, San Francisco, CA, USA, pp. 283–290 (2001)
51. Linux Community, HyperLedger fabric white paper, in *Linux Found* (2018). https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf

Publisher's Note