ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Engineering Journals
Available on IEEE Xplore®

Research Article
https://doi.org/10.1631/ENG.ITEE.2025.0043

MH-Raft: an efficient and low-latency consensus algorithm for distributed systems via MOEA/D and hybrid hierarchical clustering

Fei ZHAO, Guilong PENG, Tianyi ZANG✉

Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China

Abstract: Raft is a foundational consensus protocol for distributed systems, architected to ensure state machine replication and data consistency across machine clusters. However, traditional Raft faces significant performance bottlenecks, particularly regarding suboptimal election efficiency and substantial consensus latency in large-scale deployments. To address these challenges, this study presents MH-Raft, an enhanced consensus variant designed for high efficiency and minimal latency. We propose a hierarchical node management and election framework to optimize network coordination. Specifically, a leader election methodology leveraging the multi-objective evolutionary algorithm based on decomposition (MOEA/D) is formulated to minimize election latency by evaluating multi-dimensional node attributes. To further refine the proposed hierarchical architecture, a rigorous tightness definition is devised for optimal mediator node selection, which is integrated into a hybrid clustering algorithm that adaptively partitions the network and optimizes the mapping between mediator nodes and follower nodes. Quantitative evaluations via comprehensive experiments demonstrate that MH-Raft significantly reduces overall election latency and lowers consensus latency by 14.87%–34.45%, while enhancing average throughput by 30.43% compared to the conventional Raft implementation.

Key words: Consensus algorithm; Blockchain; Multi-objective evolutionary algorithm; Distributed systems

1 Introduction

In 2008, Satoshi Nakamoto introduced Bitcoin, which subsequently generated significant scholarly interest in blockchain technology among researchers. The foundational technology underlying Bitcoin is distributed ledger technology, with blockchain serving as one of its principal implementations. Employing cryptographic algorithms, consensus mechanisms, and smart contracts, blockchain ensures immutability, traceability, and decentralized storage of data (Ji et al., 2023; Laatikainen et al., 2023; Shi CC et al., 2023; Cheng et al., 2024; Zhao JJ et al., 2024; Eichelberger et al., 2025; Hu and Li, 2025). Blockchain exhibits substantial potential for application across diverse sectors, including privacy protection (Jia et al., 2018), copyright management (Jing et al., 2021), finance (Treleaven et al., 2017), and healthcare (Attaran, 2022), thereby fundamentally transforming conventional information technology frameworks and business models.

There are a variety of blockchain consensus algorithms, which can be divided into two categories according to the type of storage: blockchain consensus algorithms based on the chain structure (Jakobsson and Juels, 1999; Ongaro and Ousterhout, 2014; Gilad et al., 2017) and blockchain consensus algorithms based on the block structure (Bagaria et al., 2019; Kovalchuk et al., 2022; Gadiraju et al., 2023; Chen YF et al., 2024). The blockchain consensus algorithm based on the chain structure is a general form of blockchain, where each block stores the hash value of the previous block header to create a link, forming a traceable and immutable chain. In a blockchain consensus algorithm based on the block structure, a transaction can refer to multiple predecessor transactions and can be referenced by multiple subsequent transactions. Blockchain consensus algorithms based on the chain structure can be further divided into five categories, namely, proof class (Escobar et al., 2022; Lasla et al., 2022; Chatterjee et al., 2023; Zhao WB, 2023), election class (Ongaro and Ousterhout, 2014; Liu YR and Shi, 2023; Yang SJ et al., 2024), rotation class (Ben Othmen et al.,

✉ Tianyi ZANG, tianyi.zang@hit.edu.cn
Fei ZHAO, https://orcid.org/0009-0009-1555-1829
Tianyi ZANG, https://orcid.org/0000-0003-0195-8731
CLC number: TP393.02

Received: Sept. 19, 2025; Revision accepted: Mar. 22, 2026;
Crosschecked: Apr. 7, 2026

2023; Zhai et al., 2023), random class (Gilad et al., 2017; Abbasihafshejani et al., 2023), and mixed class (Bentov et al., 2014; Coelho et al., 2020). In a blockchain framework, the consensus mechanism (Castro and Liskov, 1999; Eyal et al., 2016; Chen L et al., 2017; Yang F et al., 2019; Yin et al., 2019; Keidar et al., 2021) serves as a fundamental element that facilitates agreement among all nodes regarding the authenticity of transactions, the validity of blocks, and the coherence of the ledger state. This mechanism allows numerous nodes globally to collaboratively uphold a unified, trustworthy, and continuously-updated distributed ledger without reliance on centralized authorities for validation. An effective consensus mechanism must not only guarantee the rapid and accurate generation and verification of blocks during standard operations but also ensure the system's stability and security against malicious attacks, network latencies, node failures, and other adverse circumstances. Consequently, the consensus mechanism is pivotal for the practical application and reliable functioning of blockchain technology.

The Raft algorithm (Ongaro and Ousterhout, 2014; Yang SJ et al., 2024; Shi HR et al., 2025; Sun et al., 2025) is a consensus algorithm used in distributed systems. The classic Raft algorithm belongs to the election consensus algorithm. Some algorithms combine the Raft algorithm with other consensus algorithms, which belong to the mixed consensus algorithm (Ulukök et al., 2025). The election algorithm selects a proposer by voting, which has the advantages of low computing resource requirements and high scalability (Lamport, 2019). Its disadvantages are also obvious; accounting nodes may block on multiple fork chains simultaneously. From a security perspective, the election consensus algorithm may be attacked by a malicious node to produce forged voting results. In consortium and private chains, the nodes in the network are usually known and trusted, so there is no need to rely on resource-intensive consensus mechanisms like those in public blockchains. The Raft algorithm can serve as an efficient consensus mechanism to ensure that all nodes agree on the sequence and state of transactions. In the blockchain system, the consistency of data is of vital importance. The Raft algorithm ensures data consistency across all nodes through leader election and log replication mechanisms. When new transactions or blocks are created, Raft guarantees that they are recorded and processed in an identical order on every node, thereby avoiding the problem of data inconsistency. Raft still has limitations, particularly regarding the efficiency of leader node election and the capacity for handling high loads. This study presents enhancements to the Raft algorithm through the introduction of the MH-Raft algorithm, which demonstrates superior performance relative to the original Raft algorithm in key metrics such as election latency, consensus latency, and throughput. The primary contributions of this study are as follows:

1. We introduce a leader node election methodology grounded in the multi-objective evolutionary algorithm based on decomposition (MOEA/D) optimization algorithm (Zhang and Li, 2007), which significantly decreases election latency.

2. We enhance the approach for defining tightness values, thereby facilitating the selection of superior mid-layer nodes, which are defined as mediator nodes in this study. We employ a hybrid clustering algorithm to adaptively ascertain the number of hierarchical nodes, while improving the matching process between the mediators and follower clusters.

3. We simulate multiple nodes participating in a network environment by listening to different ports on the same device and conduct experiments on synthetic node data. Results show that election latency is significantly reduced, consensus latency is reduced by 14.87%–34.45%, and throughput is increased by an average of 30.43%.

2 Related works

2.1 Research on consensus algorithms

To ensure the efficiency and security of the blockchain, consensus algorithms play a crucial role. A public chain is a permissionless chain that is open to everyone, and any node can join the blockchain network. The data on the public chain is completely transparent and accessible. Proof of work (PoW) (Jakobsson and Juels, 1999), proof of stake (PoS), and Bitcoin-next generation (Bitcoin-NG) (Eyal et al., 2016) are mainstream consensus algorithms for public chains. A consortium chain is a permissioned chain. Consortium chains require permission management and verification. Compared with public chains, they have a lower degree of openness and decentralization. Typical consensus algorithms of consortium chains include the proof of activity (PoA) (Bentov et al., 2014), practical Byzantine fault tolerance (PBFT) (Castro and Liskov, 1999), and delegated Byzantine fault tolerance (DBFT) (Wang et al., 2020). A private blockchain is a permissioned blockchain with a high degree of control and privacy. Nodes that have not been authenticated and authorized cannot join. This restriction makes the resource consumption cost of private blockchain systems relatively low. In private blockchains, the Raft consensus algorithm (Ongaro and Ousterhout, 2014) is the mainstream consensus algorithm.

2.2 Raft algorithm

In 2014, in response to the poor interpretability and difficulty of implementation of the Paxos algorithm (Lamport, 2019), Ongaro and Ousterhout (2014) proposed the Raft consensus algorithm, which defines the leader node, candidate nodes, and follower nodes. Follower nodes elect a leader node through a voting process. During the leader election phase of the Raft algorithm, when multiple candidate nodes participate, no node may receive more than half of the votes from follower nodes due to node failure or a tie, making it impossible to successfully elect a leader node. Therefore, in the event of a tie in voting, multiple rounds of voting are required to successfully select the leader node. When multiple nodes fail, the leader node cannot be selected. In Paxos, log replication typically requires negotiation in multiple stages, including the prepare stage and the commit stage. The submission of each log entry needs to go through these stages, increasing the number of communication rounds and time cost. The leader of Raft directly pushes logs to followers, reducing unnecessary negotiation steps and improving the efficiency of log replication. Raft elects leaders through a random timeout mechanism that is simple and effective, reducing the possibility of

electoral conflict. In the event of a leader failure, Raft is able to elect a new leader with an efficient election process due to the randomized timeout mechanism, ensuring system availability. Raft ensures that logs are consistent across all nodes, providing a strong consistency guarantee. Raft is widely used in many modern distributed systems, such as Etcd and Consul, which require efficient configuration management and service discovery.

2.3 Improved Raft algorithms

In the development of the Raft algorithm, Tang et al. (2022) proposed a method based on random forest classification to enhance fairness among nodes, introducing authorized nodes to reduce the probability of election failures, thereby decreasing the election time. Liu X et al. (2022) introduced a log backtracking mechanism to reduce log inconsistencies, lower the overhead of remote procedure call (RPC) communication, and improve system scalability. Yamashita et al. (2023) proposed a method to increase system throughput through batch transmission. Liu YR and Shi (2023) proposed a SHA256-based verification mechanism to enhance system security.

Yang SJ et al. (2024) proposed NK-Raft, an improved method based on the non-dominated sorting genetic algorithm II (NSGA-II) and K-means++ algorithm. The NK-Raft algorithm directly specifies the leader node using a multi-objective optimization approach, which shortens the leader election time of the original algorithm and results in better node stability. Additionally, a density value is designed to hierarchically organize the network during the log replication phase, reducing the load on the leader node and providing better scalability.

2.4 Limitations of existing approaches

Although various Raft variants have addressed specific bottlenecks, they typically treat leader election and network topology as decoupled problems. This separation leads to two major limitations: first, election optimizations often prioritize speed at the expense of leader stability; second, existing clustering methods lack the capability to adapt to dynamic node activity. Consequently, there is a critical need for a unified framework that integrates multi-objective evaluation to ensure leader quality with low complexity, while simultaneously employing dynamic, proximity-aware layering to enhance scalability.

3 Research method

3.1 Overview

In the Raft algorithm, a system node exists in one of the three states: follower, candidate, or leader. The leader node receives requests from clients and replicates the request–response pair to the majority of followers. In Raft, a log refers to an ordered sequence on the server used to record the operations performed by the state machine. When logs are synchronized to most nodes, followers are told to submit them. Followers must receive and persist the logs sent by the leader. The leader is responsible for determining when the logs are safely replicated to a majority of nodes and then informs the followers that the logs are ready to be committed. Upon receiving this notification, followers can commit the logs and apply them to their state machines. This process ensures that logs are consistently replicated across the cluster and maintains the system's reliability. A candidate is a temporary role in the leader election process. Raft requires the system to have at most one leader at any time, and only the leader and followers during normal operation.

Followers respond only to requests from other nodes. If a follower times out and does not receive a message from the leader, it becomes a candidate and starts a leader election. The candidate that receives the majority of node votes becomes the new leader. The Raft algorithm divides time into terms, and the beginning of each term is a leader election. After the leader is successfully elected, the leader manages all nodes in the cluster for the entire term. If the leader election fails, the term ends without a leader.

Although the MH-Raft algorithm also has three similar states, there are differences in the state transition. The MH-Raft algorithm has at most one leader at any time, and there are leaders, followers, and mediators during normal operation. The mediator functions as a transfer station. After receiving a request from a client, the leader synchronizes the request log to mediators. Each mediator synchronizes the request log to some followers. When logs are synchronized to most nodes, followers are told to submit logs. In Raft, the leader needs to communicate directly with all followers. The larger the network scale, the heavier the load on the leader. In MH-Raft, the leader only needs to communicate with the second-layer nodes (mediators), which greatly reduces the communication volume. Each mediator is responsible for a subset of the grouped nodes. It collects the confirmation information of the nodes within the group and reports it uniformly to the leader after more than half of the nodes have confirmed. In this way, the leader only needs to wait for the confirmation of mediators, avoiding the need to wait for responses from dozens or even hundreds of nodes. For scalability, the system can horizontally scale more nodes without affecting the performance of the leader. Newly added nodes only need to join a certain subset group, which is managed by the mediator and will not directly increase the burden on the leader.

Let the total number of nodes in the network be n. Let the number of mediators be k and $k \in [n_{\min}, n_{\max}]$, where $n_{\min}$ and $n_{\max}$ are hyperparameters to avoid extreme situations, such as all nodes being specified as mediators. There is no dependency between k and n. Instead, the specific value of k is determined through a hybrid clustering algorithm that will be discussed in detail later. During each log replication round:

1. The leader broadcasts AppendEntries to $n-1$ followers: $n-1$ messages.

2. The leader waits for at least $\lceil (n-1)/2 \rceil$ acknowledgements (ACKs): $\lceil (n-1)/2 \rceil$ messages.

Hence, in the Raft algorithm, the total number of messages per round is calculated as

$$T_{\text{Raft}}(n) = n - 1 + \left\lceil \frac{n-1}{2} \right\rceil = \mathcal{O}\left(\frac{3}{2}n\right), \tag{1}$$

where $\lceil \cdot \rceil$ denotes the ceiling function, which rounds a number up to the nearest integer. $\mathcal{O}(\cdot)$ describes how an algorithm's

running time grows as the input size increases. It usually expresses the upper bound of growth.

The remaining nodes are partitioned among k mediators, so each sub-cluster C_i $(i = 0, 1, \ldots, k-1)$ contains m_i ordinary followers, where $\sum_{i=0}^{k-1} m_i = n - k - 1$. Each round involves:

1. The leader broadcasts to k mediators: k messages.

2. Each mediator broadcasts to its sub-cluster: $\sum_{i=0}^{k-1} m_i$ messages.

3. Each mediator aggregates ACKs and replies to the leader: k messages.

The total number of messages in MH-Raft is defined as

$$T_{\text{MH-Raft}}(n) = k + \sum_{i=0}^{k-1} m_i + k = n + k - 1 = \mathcal{O}(n). \quad (2)$$

Since $k \ll n$, the total communication overhead of MH-Raft remains relatively small. The leader-side communication complexities of Raft and MH-Raft, denoted as L_{Raft} and $L_{\text{MH-Raft}}$, respectively, are shown as follows:

$$L_{\text{Raft}}(n) = \mathcal{O}(n), \quad (3)$$

$$L_{\text{MH-Raft}}(n) = \mathcal{O}(k) = \mathcal{O}(1). \quad (4)$$

If a subset group is particularly large and its m_i is very large, it will only increase the load of the mediator. For the leader, it still handles $2k$ messages, which remains unaffected by m_i. Regardless of whether k is fixed or varies with n, and regardless of whether followers are uniformly distributed or not, MH-Raft reduces the communication complexity on the leader side from $\mathcal{O}(n)$ to $\mathcal{O}(k)$. As long as $k \ll n$, a leader load reduction can be achieved. The network-wide complexity remains $\mathcal{O}(n)$, yet the leader-side communication overhead decreases from linear to constant, yielding an order-of-magnitude reduction as n increases, which fully demonstrates the advantage of the mediator mechanism. Briefly, the role of the mediator node is to share the pressure of the leader node. The state transition diagrams for the Raft and MH-Raft algorithms are shown in Fig. 1.

We illustrate the specific steps of MH-Raft in Fig. 2. We list the methods used and the participants involved in each step, and provide a visual representation of all steps. In the MH-Raft algorithm, in the first step, we use a multi-objective optimization method based on MOEA/D (Zhang and Li, 2007) to elect the leader node. In the second step, we design an improved definition of tightness value, which is combined with a hybrid clustering algorithm designed by us to determine the number of mediator nodes. We provide a formulaic definition in the following subsection. In the third step, we design an adaptive clustering algorithm to cluster the remaining follower nodes. In the fourth step, follower node groups and mediator nodes are matched to complete the log synchronization process. We provide a detailed description of each step along with some theoretical derivation.

3.2 Leader election based on MOEA/D

Previous research on the selection of the leader node using multi-objective optimization techniques has predominantly employed the NSGA-II algorithm, which typically selects only 2–3 attributes of a node. However, such low-dimensional representations may inadequately capture the characteristics of nodes. It is posited that a more comprehensive representation of nodes can be achieved through the inclusion of additional data attributes, thereby enhancing the stability of the selected leader node. In this study, we identify six attributes to constitute the node data: node load level, frequency of down-

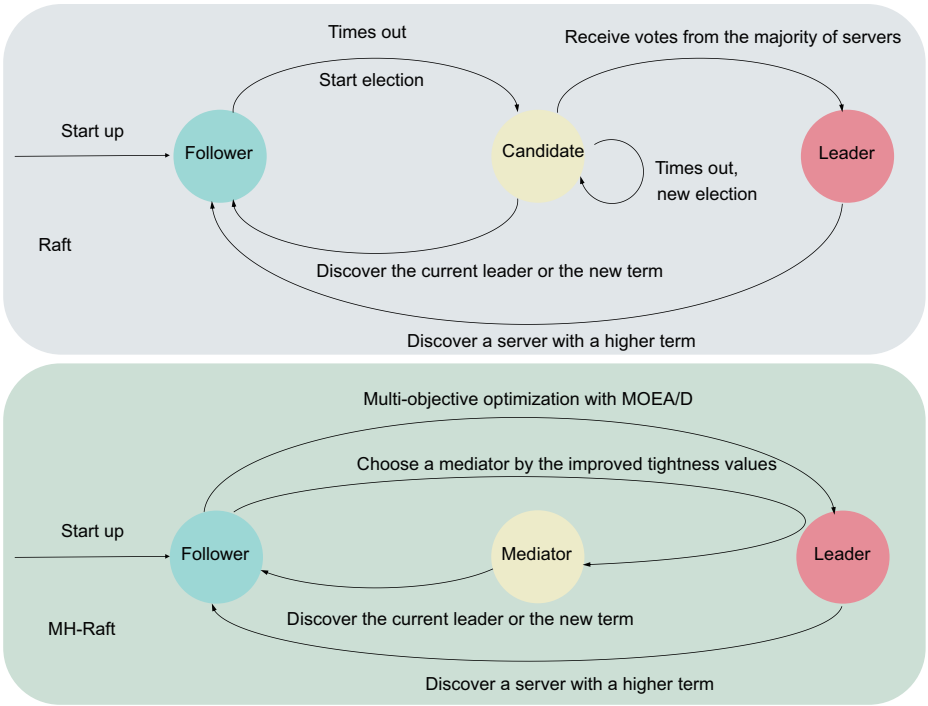

Fig. 1 State transition diagrams for Raft and MH-Raft

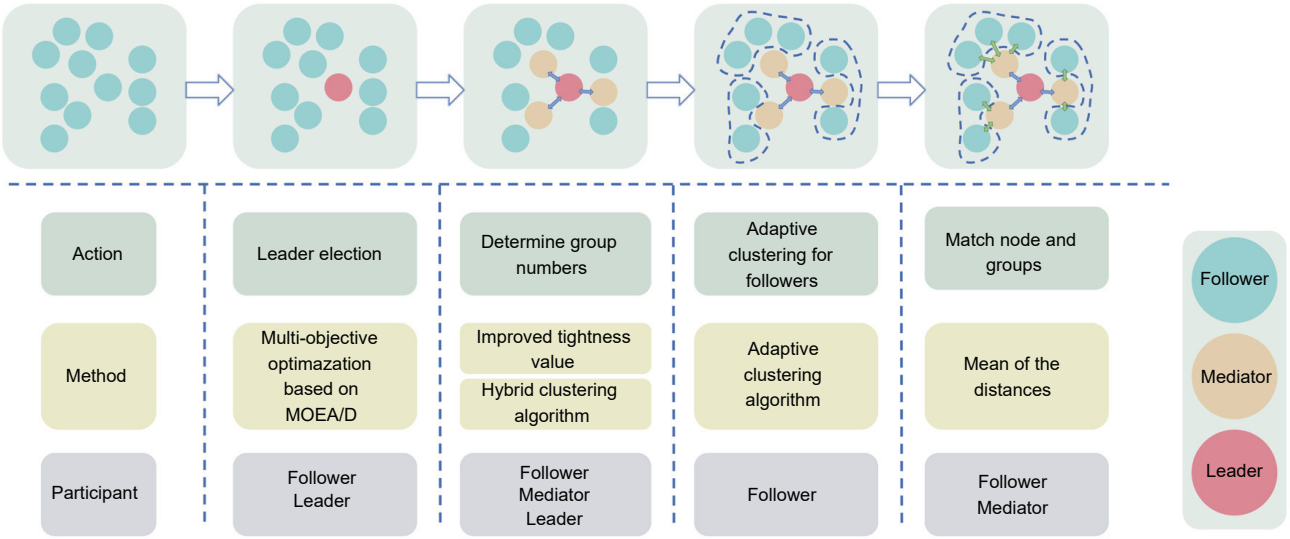

Fig. 2 Flow of the MH-Raft algorithm

times, online hours, communication costs, storage resource utilization, and creation time. The first three attributes are used in NK-Raft and remain unchanged here. For the last three attributes, nodes with lower communication costs, higher storage resource utilization, and shorter creation times are more suitable to be the leader node.

As the dimensionality of data and the number of nodes increase, NSGA-II's convergence rate tends to diminish significantly. Consequently, we opt to employ the MOEA/D optimization algorithm for the selection of the leader node. We compare the complexity of the two algorithms and define N as the number of individuals, D as the dimension, G as the number of iterations, P_c as the crossover probability, P_m as the mutation probability, H as the number of sub-problems, and T as the neighborhood size. The individual evaluation complexity of both NSGA-II and MOEA/D is $\mathcal{O}(ND)$. The cross-mutation complexity of NSGA-II is $\mathcal{O}(GN(P_\mathrm{c}D + P_\mathrm{m}D))$ and the reproductive complexity of MOEA/D is $\mathcal{O}(GHD)$. When the number of nodes is large, $P_\mathrm{c}N + P_\mathrm{m}N$ is much larger than H. Therefore, as N and D increase, the gap in complexity between the two algorithms will gradually widen. The fast non-dominated sorting complexity of NSGA-II is $\mathcal{O}(GN^2)$, which is quadratic with respect to N, whereas the complexities of the decomposition process and neighborhood update in MOEA/D are $\mathcal{O}(GNHM + GHTM)$, which is polynomial related to N. Obviously, the complexity of MOEA/D increases more slowly as N increases. This implies that when we select more node attributes, the MOEA/D algorithm is more suitable for the multi-objective optimization process in the election stage than the NSGA-II algorithm.

3.2.1 Multi-objective optimization and MOEA/D

The constrained multi-objective optimization problem can be expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} F(\boldsymbol{x}) &= [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})] \\ \text{s.t. } & g_i(\boldsymbol{x}) \geq 0,\ i = 1, 2, \ldots, M, \\ & h_j(\boldsymbol{x}) = 0,\ j = 1, 2, \ldots, L, \end{aligned} \tag{5}$$

where $\boldsymbol{x}$ denotes the decision vector, which represents the variables to be optimized in multi-objective optimization problems. In our scenario, it represents the node selection outcome. F denotes the multi-objective optimization function, f_i signifies the i^{th} objective optimization function, g and h represent the constraints associated with the optimization problem, M represents the number of inequality constraints, and L represents the number of equality constraints. During the election phase of the MH-Raft algorithm, each node is characterized by six distinct attributes, which can be translated into their respective objective functions. The selected attributes are independent of one another, resulting in a multi-objective optimization problem with six objective functions (i.e., $K = 6$). Constraints are established and articulated based on the characteristics of each attribute, thereby framing the problem as a multi-objective optimization scenario as outlined in Eq. (5).

The MOEA/D algorithm reformulates multi-objective optimization challenges into several scalar subproblems by using a divide-and-conquer strategy. In the optimization of these subproblems, each one leverages information from its neighboring counterparts, which contributes to a reduction in the computational complexity during each generation when compared to NSGA-II, particularly in scenarios involving a large number of objective functions. The solution selection process within MOEA/D is predicated on a decomposition approach, which identifies superior solutions for the weight vectors based on the aggregated values of the objective functions. Given that our optimization functions are relatively straightforward and consist entirely of convex functions, we opt for a weighted sum method, as delineated in Eq. (6):

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}|\boldsymbol{\lambda}) = \sum_{i=1}^{K} \lambda_i f_i(\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_K]^{\mathrm{T}}$ is the weight vector, satisfying $\lambda_i \geq 0$ and $\sum_{i=1}^{K} \lambda_i = 1$. Since the node attributes are used directly as the objective function, each single-objective optimization function is characterized as a convex function.

This approach is capable of identifying all Pareto optimal solutions, thereby facilitating the identification of the optimal leader nodes.

3.2.2 Leader election

The election phase of the MH-Raft algorithm is illustrated in Fig. 3. Initially, an objective function is formulated based on node attributes, incorporating relevant constraints. Subsequently, the MOEA/D algorithm is employed to obtain the optimal node, designated as the leader node, with the second-best node serving as a backup. If the optimal node fails, the second-best node is elevated to the position of leader to ensure the continuity of normal operations.

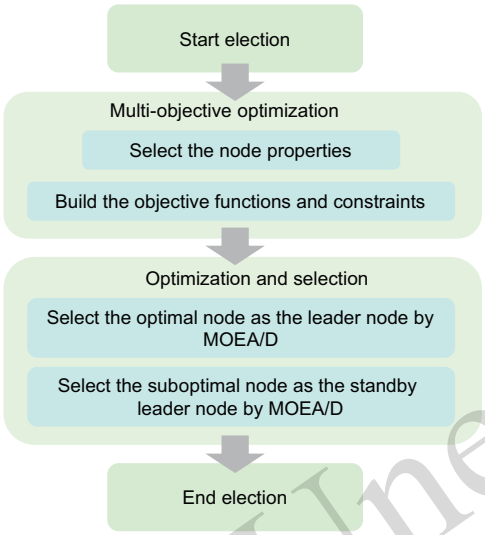

Fig. 3 Leader election of MH-Raft

3.3 Improved tightness values

The definition of the tightness value presented in prior research is characterized by its simplicity and, as a result, is incapable of adequately capturing the intricate interrelationships among nodes. This study aims to improve the definition of the tightness value to more accurately represent the relationship between follower nodes and the leader node, thereby facilitating the selection of more optimal mediator nodes. The application of the Manhattan distance metric, derived from latitude and longitude, is insufficiently precise for measuring geographical distance DN_i. Therefore, the Haversine formula is employed to accurately compute the true geographical distance, as captured by the corresponding mathematical expressions:

$$\mathrm{DN}_i = -d_i, \tag{7}$$

$$d_i = 2R \arcsin \sqrt{\sin\left(\frac{\Delta\phi_i}{2}\right)^2 + \cos\phi_0 \cos\phi_i \sin\left(\frac{\Delta\gamma_i}{2}\right)^2}, \tag{8}$$

$$\Delta\phi_i = \phi_i - \phi_0, \tag{9}$$

$$\Delta\gamma_i = \gamma_i - \gamma_0, \tag{10}$$

where ϕ_0 and γ_0 denote the latitude and longitude of the leader node, while ϕ_i and γ_i denote the latitude and longitude of the i^{th} follower node, respectively. R is the radius of the Earth. As the spatial separation between follower nodes and the leader node expands, the latency associated with data transmission escalates, resulting in increased communication latency among nodes and a heightened packet loss rate. These factors contribute to the diminished suitability of geographically distant nodes as mediators. The negative coefficient in Eq. (7) serves to calibrate the variable in relation to the tightness value, indicating that a shorter geographical distance correlates with a higher tightness value. Node activeness serves as a critical determinant influencing consensus latency. Nodes exhibiting elevated levels of activeness are more apt to function as mediators. Consequently, we maintain the node activeness variable, which is defined as

$$\mathrm{AN}_i = \frac{\mathrm{OT}_i}{\mathrm{ST}_i}, \tag{11}$$

where AN_i, OT_i, and ST_i denote the activeness, online duration, and the duration since being added to the blockchain of the i^{th} node, respectively. We seek nodes with local centrality and reliability, which directly determine the communication success rate of the hierarchical structure. The initial definition of the tightness value was established as a linear combination of the Manhattan distance, derived from latitude and longitude, and the online duration ratio. It is important to note that these two variables incorporated in this formula are not normalized, which could result in one attribute, presumably the distance, exerting an excessively large influence on the overall calculation. Consequently, this study proposes a revised definition of the tightness value as follows:

$$\mathrm{TN}_i = w_1 \frac{\mathrm{DN}_i}{\sum_{j=0}^{k-1} \mathrm{DN}_j} + w_2 \frac{\mathrm{AN}_i}{\sum_{k=0}^{k-1} \mathrm{AN}_k}. \tag{12}$$

Incorporating normalization techniques to equilibrate these two influencing factors enhances the sensitivity of the independent variable's effect on the dependent variable to the weights w_1 and w_2. Typically, these weights are initialized at a value of 0.5; however, they can be adjusted according to specific scenarios.

3.4 Hybrid clustering algorithm for node grouping

In accordance with methodologies established in prior research (Yang SJ et al., 2024), the n nodes exhibiting the highest tightness values are designated as mediator nodes. The remaining follower nodes are subsequently organized into n groups, where each group is managed by a single mediator. The parameter n is regarded as a hyperparameter, conventionally set to 4. However, as the total number of nodes increases, the efficacy of maintaining only four mediator nodes in the second layer may diminish. Consequently, the determination of the appropriate number of mediator nodes emerges as a pertinent issue. This study proposes a hybrid clustering approach to ascertain the number of groups, which subsequently informs the selection of the number of mediator nodes. Ultimately, we identify a suitable clustering algorithm to facilitate the clustering of the mediator node count. A flowchart illustrating this process is presented in Fig. 4.

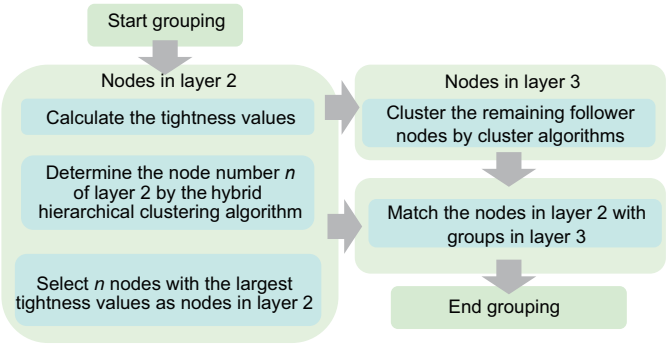

Fig. 4 Flowchart of node grouping

3.4.1 Determination of group numbers

We design a hybrid clustering algorithm to determine the number of mediator nodes, specifically to determine the number of groups for other follower nodes. The specific algorithm process is as follows:

1. Set the minimum group number $k_{\min}$ and the maximum group number $k_{\max}$;

2. Apply the agglomerative hierarchical clustering method (Sokal and Michener, 1958) to all follower nodes based on geographical location, and obtain the number of groups k_0 when the algorithm stops;

3. Apply the divisive hierarchical clustering method (Ward, 1963) to all follower nodes based on geographical location, and obtain the number of groups k_1 when the algorithm stops;

4. The number of final groups is defined as $k = G(k_{\min}, k_{\max}, k_0, k_1)$.

The function G is represented as

$$G = \frac{1}{2}(\max(k_{\min}, k_0) + \min(k_{\max}, k_1)). \tag{13}$$

Setting minimum and maximum cluster bounds ensures the final cluster count remains within a reasonable range. Otherwise, when the number of mediator node clusters is too large or too small, the MH-Raft algorithm may degenerate into the Raft algorithm. After obtaining the final number of clusters n, the n nodes with the highest tightness values are selected as mediator nodes, while the remaining nodes serve as follower nodes.

We employ a hybrid hierarchical clustering algorithm to determine the number of mediator nodes, so the number of clusters cannot be predetermined as a stopping criterion. The hierarchical clustering algorithms used in this study all employ dynamic stopping criteria. Taking the agglomerative hierarchical clustering algorithm as an example, the merging process stops when the similarity after merging two clusters falls below a certain threshold. The pseudo-codes for the agglomerative hierarchical clustering algorithm and divisive hierarchical clustering algorithm used in our algorithm are shown in Algorithms 1 and 2, respectively.

Through the above process, we are able to determine the number of mediator nodes, as well as the number of groups after grouping follower nodes in the third layer. In the next section, we will adaptively select a clustering algorithm and group follower nodes in the third layer based on the process of determining the number of mediator nodes in the second layer.

Algorithm 1 Agglomerative hierarchical clustering algorithm

Input: Dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_m\}$ (x_i denotes the i^{th} node participating in the clustering algorithm, m is the total number of participating nodes) and similarity threshold ϵ
Output: Groups $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$

Initialize each node as a separate group: $\mathcal{G} \leftarrow \{\{x_1\}, \{x_2\}, \ldots, \{x_m\}\}$
for all $G_i, G_j \in \mathcal{G}, i \neq j$ **do**
  $\text{dist}_{\min} \leftarrow \infty$ /* $\text{dist}_{\min}$ represents the minimum Euclidean distance */
  for all $x_p \in G_i$, $p = 1, 2, |G_i|$ **do**
    for all $x_q \in G_j$, $q = 1, 2, |G_j|$ **do**
      $\text{dist} \leftarrow d(x_p, x_q)$ /* dist is the Euclidean distance */
      if $\text{dist} < \text{dist}_{\min}$ **then**
        $\text{dist}_{\min} \leftarrow \text{dist}$
      end if
    end for
  end for
  $M_{ij} \leftarrow \text{dist}_{\min}$
end for
while True **do**
  Find the pair of groups (G_a, G_b) with the highest similarity: $\max_{G_i, G_j \in \mathcal{G}, i \neq j} M_{ij}$ /* M_{ij} indicates the minimum distance between node sets G_i and G_j */
  if $\max M_{ij} < \epsilon$ **then**
    break
  else
    Merge G_a and G_b: $G_{\text{new}} \leftarrow G_a \cup G_b$
    Update $\mathcal{G}$: $\mathcal{G} \leftarrow (\mathcal{G} \setminus \{G_a, G_b\}) \cup \{G_{\text{new}}\}$
    for all $G_i \in \mathcal{G} \setminus \{G_{\text{new}}\}$ **do**
      $\text{dist}_{\min} \leftarrow \infty$
      for all $x_p \in G_{\text{new}}$ **do**
        for all $x_q \in G_i$ **do**
          $\text{dist} \leftarrow d(x_p, x_q)$
          if $\text{dist} < \text{dist}_{\min}$ **then**
            $\text{dist}_{\min} \leftarrow \text{dist}$
          end if
        end for
      end for
      $M_{i\text{new}} \leftarrow \text{dist}_{\min}$
    end for
  end if
end while
return $\mathcal{G}$

Quantitatively, the traditional Raft algorithm imposes a linear communication overhead of $\mathcal{C}_{\text{Raft}} \approx O(n)$ on the leader, creating a significant processing bottleneck as the network size n expands. In contrast, MH-Raft's layered architecture mitigates this limitation by restricting the leader's direct interaction to k proxy nodes, effectively reducing the leader's load to $\mathcal{C}_{\text{MH}} \approx O(k)$. By distributing the remaining replication burden to proxy nodes that operate in parallel with a local complexity of $O(n/k)$, and given that $k \ll n$, MH-Raft fundamentally transforms the log replication process from a linear to a sub-linear growth model, providing a rigorous theoretical justification for the enhanced scalability and throughput observed in our experiments.

3.4.2 Adaptive clustering algorithm

We use clustering algorithms to cluster the remaining follower nodes into groups, with the number of groups equal to the number of mediator nodes in the second layer (Algorithm 3).

Algorithm 2 Divisive hierarchical clustering algorithm

Input: Dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_m\}$ and similarity threshold ϵ
Output: Groups $\mathcal{G} = \{G_1, G_2, \ldots, G_k\}$

Initialize all nodes as a single group: $\mathcal{G} \leftarrow \mathcal{D}$
for all $G \in \mathcal{G}$ **do**
  $\text{dist}_{\max} \leftarrow 0$ /* $\text{dist}_{\max}$ represents the maximum Euclidean distance */
  for all $x_p \in G$ **do**
    for all $x_q \in G$, $x_q \neq x_p$ **do**
      $\text{dist} \leftarrow d(x_p, x_q)$
      if $\text{dist} > \text{dist}_{\max}$ **then**
        $\text{dist}_{\max} \leftarrow \text{dist}$
      end if
    end for
  end for
  $\text{similarity}(G) \leftarrow \text{dist}_{\max}$
end for
while True **do**
  Find the group G_{split} with the lowest internal similarity
  Let $S_{\min}$ be the minimum internal similarity of G_{split}
  if $S_{\min} > \epsilon$ **then**
    break
  else
    Split G_{split} into two new groups G_{new1} and G_{new2}
    Update $\mathcal{G}$: $\mathcal{G} \leftarrow (\mathcal{G} \setminus \{G_{\text{split}}\}) \cup \{G_{\text{new1}}, G_{\text{new2}}\}$
    for $G_{\text{new}} \in \{G_{\text{new1}}, G_{\text{new2}}\}$ **do**
      $\text{dist}_{\max} \leftarrow 0$
      for all $x_p \in G_{\text{new}}$ **do**
        for all $x_q \in G_{\text{new}}$, $x_q \neq x_p$ **do**
          $\text{dist} \leftarrow d(x_p, x_q)$
          if $\text{dist} > \text{dist}_{\max}$ **then**
            $\text{dist}_{\max} \leftarrow \text{dist}$
          end if
        end for
      end for
      $\text{similarity}(G_{\text{new}}) \leftarrow \text{dist}_{\max}$
    end for
  end if
end while
return $\mathcal{G}$

Algorithm 3 Adaptive clustering algorithm

Input: $k_{\min}$, $k_{\max}$, k_0, and k_1
if $\max(k_0, k_1) = k_0$ and $k_0 \in (k_{\min}, k_{\max})$ **then**
  Use the agglomerative hierarchical clustering algorithm as the final clustering algorithm
else if $\max(k_0, k_1) = k_1$ and $k_1 \in (k_{\min}, k_{\max})$ **then**
  Use the divisive hierarchical clustering algorithm as the final clustering algorithm
else
  Use the K-means++ clustering algorithm as the final clustering algorithm
end if
return the selected clustering algorithm

The definitive number of clusters is determined by the hybrid algorithm, which is subsequently used to classify other nodes. Specifically, if the agglomerative hierarchical clustering algorithm yields the highest number of groups, this algorithm is applied to cluster the remaining follower nodes.

Conversely, if the divisive hierarchical clustering algorithm produces the greatest number of groups, it is selected for the clustering of the other follower nodes. In instances where the number of groups from both methods falls outside the predetermined minimum and maximum thresholds, the K-means++ algorithm is implemented for clustering purposes.

3.4.3 Matching mediator nodes and follower groups

Upon grouping the follower nodes within the third layer, it becomes imperative to match the mediator nodes of the second layer with the corresponding groups of the third layer. This step facilitates communication between the mediator nodes in the second layer and those within the matched groups of the third layer. In the context of the NK-Raft algorithm, the Euclidean distance from the mediator nodes in the second layer to the centroids of the groups in the third layer is computed, with the node exhibiting the shortest distance being designated as the match for the respective group. However, if the distribution of follower nodes is asymmetric or contains multiple aggregation centers, the centroid of the follower groups may not accurately reflect the true location characteristics. Therefore, a more representative measure of proximity between the nodes and the groups can be achieved by calculating the average distance from the mediator nodes in the second layer to follower nodes within each group of the third layer.

Although this approach necessitates the computation of a greater number of distances among nodes, thereby increasing complexity, the current scale of nodes does not warrant serious consideration of this complexity. Consequently, to match the nodes and groups more appropriately, we advocate the adoption of the average distance calculation method (Algorithm 4). The difference in distance calculation between NK-Raft and MH-Raft can be shown in Fig. 5. In fact, the matching results produced by this distance calculation method are almost consistent with the clustering results of selecting the second-layer nodes. Therefore, given the high computational overhead, we can use the clustering result directly as a substitute for the exact matching result when determining the number of mediator nodes.

Algorithm 4 Matching algorithm for mediator nodes and follower groups

Input: Mediator nodes $\mathcal{M} = \{M_i | i = 1, 2, \ldots, k\}$ and follower groups $\mathcal{G} = \{\{F_r\} | F_r \in G_j,\ j = 1, 2, \ldots, n,\ r = 1, 2, \ldots, |G_j|\}$

for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    for $F_r \in G_j$ **do**
      Calculate the Euclidean distance of mediator M_i and follower F_r: $D(M_i, F_r)$
    end for
    Calculate the distance of mediator M_i and follower group G_j: $D(M_i, G_j) = \frac{1}{|G_j|} \sum_{F_r \in G_j} D(M_i, F_r)$
  end for
  Matching the mediator M_i with the follower group that has the minimum distance $(M_i, G_{\min})$
  $(\mathcal{G} \leftarrow \mathcal{G} \setminus \{G_{\min}\})$
end for
return Matching pairs $(\mathcal{M}, \mathcal{G})$

4 Experiments and analysis

We implement the Raft algorithm and the improved MH-Raft algorithm using the Go programming language. We simulate multiple nodes participating in a network environment by

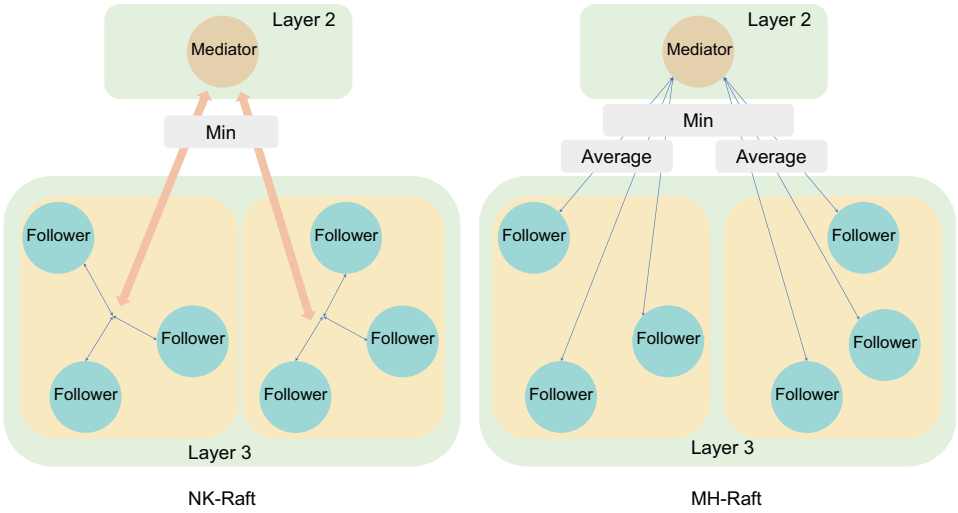

Fig. 5 Difference in distance calculation between NK-Raft and MH-Raft

listening on different ports on the same device, and we test the election latency, consensus latency, and throughput of the algorithms. The experimental environment of MH-Raft is shown in Table 1.

Table 1 Experimental environment of MH-Raft

Configuration	Detail
Operating system	Windows 11, 64-bit
CPU	13$^{\text{th}}$ Gen Intel® Core™ i9-13905H@2.60 GHz
RAM	32.0 GB
Disk	1.86 TB
Go version	1.24.1

Please note that our experimental data are artificially generated rather than real-world data. The attributes of each node are generated by random numbers. We conduct 10 experiments on each algorithm and present the statistical results.

Although we simulate multiple nodes by listening to different ports on a single machine through the Go programming language, which is indeed different from a wide area network (WAN), this setup is one of the standard practices for verifying the logical efficiency of protocols in consensus algorithm research. Isolating algorithm logic consumption: single-machine experiment eliminates uncontrollable physical link jitter, enabling us to measure more purely the performance changes brought by the MH-Raft algorithm itself compared to traditional Raft. Relative performance effectiveness: our core claim is that MH-Raft reduces the message complexity and election rounds compared to Raft and NK-Raft. Whether in a local area network or a WAN, a reduction in the number of communications will inevitably lead to a decrease in latency. Therefore, although the absolute value may differ from the real WAN, the trend and ratio of performance improvement are externally valid.

The experimental settings are shown in Table 2. As the number of nodes increases, the population size of either MOEA/D or NSGA-II should also increase accordingly. Therefore, we directly set the population size to the number of nodes. The two weight hyperparameters w_1 and w_2 are selected to reflect the effect of the MH-Raft algorithm's normalization operation when the tightness value is improved; that is, the influence of each variable on the tightness value is only controlled by the coefficient. The selection of $k_{\min}$ is designed according to the minimum number of nodes, which is 5 in our experiments, meaning that four nodes are remaining after all nodes elect a leader node. The setting of $k_{\max}$ takes into account the extreme situation where each follower group has only one node. Since it is a boundary hyperparameter, the range can be widened. The similarity threshold parameter is involved in the hierarchical clustering algorithm, and its setting is related to the data points. We hope that under the set threshold, the number of clustering groups generated by the hierarchical clustering algorithm is as close as possible to the range between $k_{\min}$ and $k_{\max}$. The values 0.15 and 0.25 are thresholds suitable for our simulated data.

Table 2 Experimental settings

Hyperparameter	Value
Number of generations	50
w_1	0.50
w_2	0.50
$k_{\min}$	4
$k_{\max}$	$\lfloor \frac{n-1}{3} \rfloor$
ϵ_1	0.15
ϵ_2	0.25
Metric method	Euclidean distance

$\lfloor \cdot \rfloor$ represents the floor function

4.1 Leader election latency

In the Raft consensus algorithm, a new election needs to be held when a leader node fails. The time taken for the re-election is referred to as the election latency:

$$T_{\text{election}} = t_{\text{leader}} - t_{\text{mal}}, \tag{14}$$

where T_{election}, t_{leader}, and t_{mal} denote the election latency, the moment when a new leader node is elected, and the moment

when the old leader node fails, respectively. In this section, we conduct experiments with 5–30 nodes, comparing the election latency of Raft, NK-Raft, and MH-Raft. The results are shown in Fig. 6.

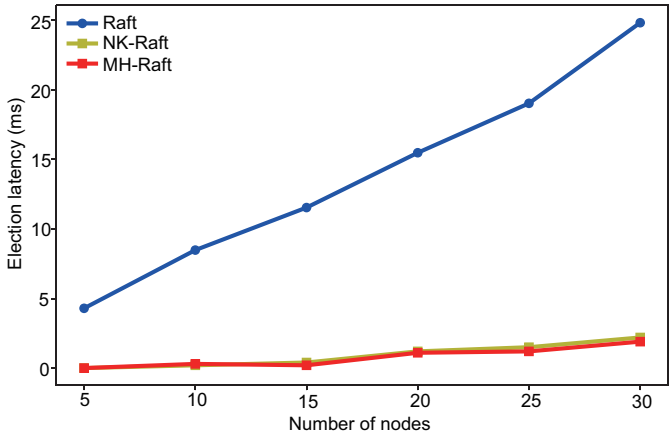

Fig. 6 Leader election latency of Raft, NK-Raft, and MH-Raft

It can be seen that as the number of nodes increases, the election latency of both Raft and MH-Raft rises. However, the election latency of MH-Raft is consistently lower than that of the Raft algorithm. MH-Raft conducts elections through a multi-objective optimization algorithm. When the number of nodes increases, the time cost required by the MOEA/D algorithm does not increase significantly. Experimental results show that the election latency of both MH-Raft and NK-Raft (which uses the NSGA-II algorithm as the optimization algorithm) remains at a consistently low level. However, due to the slightly higher time cost of the NSGA-II algorithm in high dimensions than the MOEA/D algorithm, the overall performance shows that the election latency of NK-Raft is slightly higher than that of MH-Raft.

Fig. 7 presents the errors of MH-Raft and NK-Raft, using the sample standard error (SE) $S_{\overline{X}}$ as the error measurement criterion, which is defined as

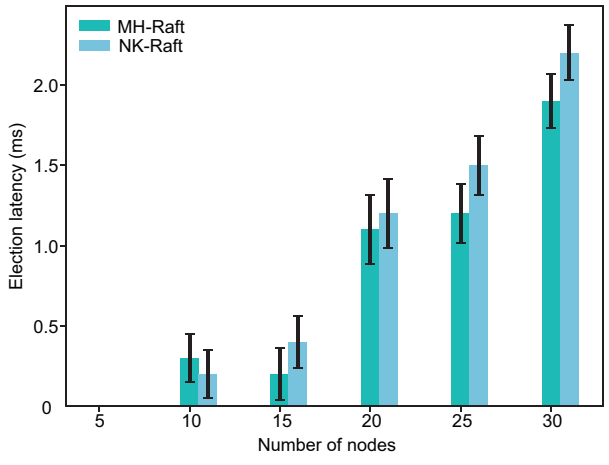

Fig. 7 SEs for the leader election latency of NK-Raft and MH-Raft

$$S_{\overline{X}} = \frac{S}{\sqrt{n_e}}, \tag{15}$$

$$S = \sqrt{\frac{\sum_{i=1}^{n_e}(X_i - \overline{X})^2}{n_e - 1}}, \tag{16}$$

where n_e is the number of experiments. X_i and $\overline{X}$ represent the leader election latency obtained from the i^{th} experiment and the mean value of all experiments, respectively. Lower election latency means that the system is able to elect a new leader more quickly in the event of a leader failure, reducing the time that the system is leaderless and improving system availability. This is especially important in high-load and high-concurrency scenarios, where any latency can cause performance degradation or even a crash.

In addition, the Raft algorithm's election process requires a certain amount of system resources, especially network bandwidth and computing resources. The election process based on the MOEA/D algorithm applied in MH-Raft does not require communication between nodes, so it consumes less computing resources and negligible network resources, thus saving system resources.

4.2 Consensus latency

Consensus latency is the time required for a client to send a request and receive a response, represented by

$$T_{\text{consensus}} = t_{\text{receive}} - t_{\text{send}}, \tag{17}$$

where $T_{\text{consensus}}$, t_{receive}, and t_{send} denote the consensus latency, the moment when the client receives the response, and the moment when the client sends the request, respectively. In this experiment, the number of transactions contained in each block is fixed at 1000, and the number of nodes participating in the consensus ranges from 5 to 30. The average of multiple experimental data points is taken as the experimental result for consensus latency when comparing MH-Raft with Raft and NK-Raft. These results are shown in Fig. 8. As the node scale increases from 5 to 30, the consensus latency of MH-Raft is 37.2, 72.9, 103.1, 133.4, 142.4, and 154.3 ms, while the consensus latency of Raft is 43.7, 81.8, 121.2, 160.9, 213.0, and 235.4 ms. The reduction in latency is between 14.87% and 34.45%.

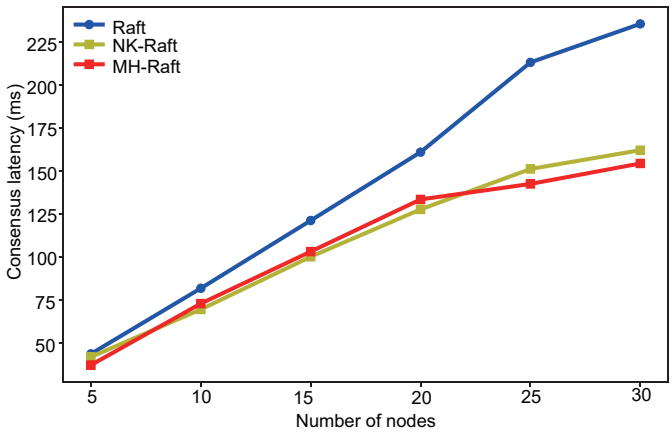

Fig. 8 Consensus latencies of Raft, NK-Raft, and MH-Raft

From Fig. 8, note that as the number of nodes increases, the consensus latencies of Raft, NK-Raft, and MH-Raft increase, but the consensus latency of MH-Raft is lower than that of Raft. The above results indicate that MH-Raft has lower consensus latency compared to Raft, thereby saving communication time overhead. Fig. 9 presents the errors of MH-Raft and NK-Raft, using the sample SE $S_{\overline{X}}$ as the error measurement criterion. The adaptive clustering algorithm keeps the number of mediator nodes within an appropriate range instead of fixing it at four. This method reduces the pressure on the mediator nodes while preventing their number from being too large. This is the fundamental reason for the low consensus latency. Combined with the previous experiments, the fundamental reason for the increase in throughput is the reduction in consensus latency, which shortens the time required for processing each transaction.

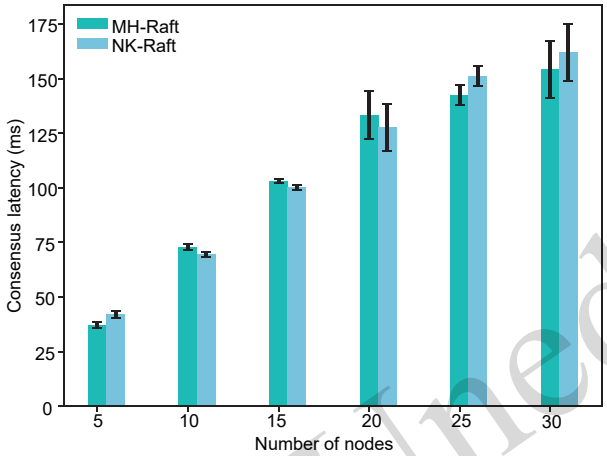

Fig. 9 SEs for the consensus latency of NK-Raft and MH-Raft

Lower consensus latency ensures that log entries are replicated more quickly to most nodes, improving data consistency. This helps reduce the possibility of data inconsistencies and ensures that the data are kept synchronized across all nodes. The lower consensus latency makes the log replication process faster and more reliable, which helps simplify the maintenance and troubleshooting process of the system. It is worth noting that the application of the clustering algorithm may lead to an increase in the consensus latency, but experimental results indicate that the increased latency is much smaller than the reduction achieved by applying mediators because the consensus latency is maintained at a lower level as a whole in the experiments.

4.3 Throughput

Throughput refers to the number of transactions that can be processed in a unit time. The higher the throughput, the better the system's ability to handle transaction requests. It is usually calculated by the number of transactions per second, as shown in Eq. (18):

$$R_{\mathrm{tps}} = \frac{N_{\mathrm{Tr}}}{\Delta t}, \tag{18}$$

where Δt denotes the duration from the moment a transaction is packed until the block is confirmed. N_{Tr} denotes the number of transactions packed in the duration. In these experiments, the number of transactions contained in each block is fixed at 1000 and the number of nodes participating in the consensus ranges from 5 to 30. The average value of multiple experimental data is taken as the throughput result to compare MH-Raft with Raft. Experimental results are shown in Fig. 10. As the node scale increases from 5 to 30, the throughput of MH-Raft is 27218, 13758, 9705, 7805, 7098, and 6785 transactions/s, while the throughput of Raft is 22955, 12255, 8278, 6239, 4726, and 4262 transactions/s. The averge increase of throughput is 30.43%.

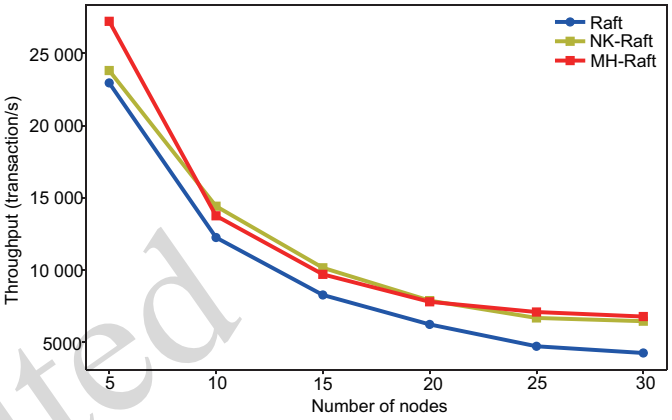

Fig. 10 Throughput of Raft, NK-Raft, and MH-Raft

Note that both Raft and MH-Raft experience a decrease in throughput as the number of nodes increases; however, the throughput of MH-Raft is consistently slightly higher than that of Raft. Fig. 11 presents the errors of the MH-Raft and NK-Raft algorithms, using the sample SE, denoted as $S_{\overline{X}}$, as the error measurement criterion. High throughput means that the system can process a large number of requests quickly and improve resource utilization, thereby enhancing the overall system performance. At the same time, it can reduce system pressure under high load, improving the stability of the system. This helps mitigate the risk of system crashes and ensures that the system can function under high loads. Experiments demonstrate that the throughput of the MH-Raft algorithm is higher than that of the Raft algorithm with different node numbers, and it is more likely to show advantages in high-load

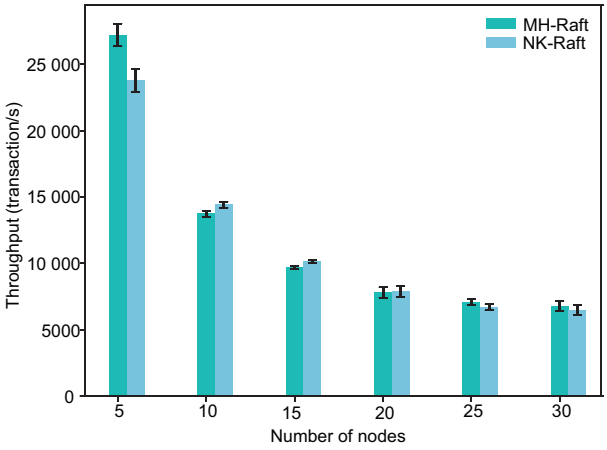

Fig. 11 SEs for the throughput of NK-Raft and MH-Raft

scenarios.

The overall throughput improvement observed in our evaluation is a combined result of two factors. Primarily, the proposed multi-objective optimized election mechanism significantly reduces the system's downtime during leader transitions, allowing the network to resume transaction processing much faster. Secondarily, the hierarchical topology inherently benefits the log replication phase. By using mediators for log dissemination, the direct broadcast burden and network input/output (I/O) bottleneck on the main leader are alleviated compared to the flat structure of standard Raft. While the current experimental setup evaluates the system's macro-level performance where these two effects are coupled, conducting a decoupled, fine-grained analysis to isolate the exact throughput gains from replication enhancements remains a valuable direction for our future work.

4.4 Hybrid clustering algorithm

The selection of the adaptive algorithm is shown in Table 3, as recorded in the experiments. Specifically, we record how the algorithm conducts clustering in each experiment. It can be seen that when the number of nodes gradually increases, most of the adaptive clustering algorithms use the agglomerative clustering algorithm as the clustering algorithm. When the number of nodes is small, the adaptive clustering algorithm adopts the K-means++ algorithm, which is in line with our expectation. If the K-means++ algorithm is not set, the appropriate clustering algorithm cannot be selected when the number of nodes is 5, and the two hierarchical clustering algorithms are not effective when the number of nodes is 10. The adaptive clustering algorithm successfully avoids these two cases and chooses the K-means++ algorithm we set as the clustering algorithm. The occurrence of this situation shows that our algorithm is quite robust.

Table 3 Selection of the adaptive algorithm

Number of nodes	Most selected algorithm
5	K-means++
10	K-means++
15	K-means++
20	Hierarchical clustering
25	Hierarchical clustering
30	Hierarchical clustering

5 Case study

According to the China Health Statistical Yearbook (2024), there are over 8700 public hospitals at the secondary level and above requiring nationwide medical data sharing. Constructing a unified blockchain for such a massive scale (8000+ nodes) poses significant challenges to traditional flat consensus protocols, particularly regarding convergence latency and network instability. To address this, a five-layer hierarchical networking model is designed to bridge national-level consensus with provincial and local institution layers. To verify the scalability of MH-Raft in a representative subscenario of this larger network, we take a 30-node regional medical consortium chain as an example, using the experimental data in Section 4 of this study.

The regional medical consortium chain demands "second-level confirmation, automatic failover, and low cost." MH-Raft addresses these requirements by reducing the leader election latency, ensuring that the core enterprise's front-end confirmation page remains unnoticeable during planned maintenance. Moreover, the mediator mechanism reduces the leader-side communication complexity and monthly cloud-traffic costs, thereby directly fulfilling the low-cost requirement. The distribution of roles in the scenario is as follows: one core top-tier hospital, eight secondary hospitals, 12 primary health centers, six medical insurance/commercial insurance institutions, and three regional laboratory/imaging centers. The topology is a two-layer star: layer 1 consists of the leader and mediators and layer 2 comprises each mediator star-connected to followers. This single-host, multi-port, two-layer star configuration is designed for algorithm evaluation. Experimental results for the 30-node scenario show that the leader election latency is reduced from 24.8 ms (Raft) to 1.9 ms (MH-Raft), the consensus latency is reduced from 235.4 ms (Raft) to 168.7 ms (MH-Raft), and the peak throughput is increased from 23 800 transactions/s (Raft) to 32 000 transactions/s (MH-Raft). Meanwhile, the 1.9-ms failover enables the planned maintenance of the original equipment manufacturer to be imperceptible to the front-end rights confirmation. The 0.17-s transaction confirmation meets the second-level receipt commitment of medical consortium institutions. The 32 000 transactions/s margin can cover the 220 transactions/s demand of a future daily peak of 800 000 transactions. In the scenario of a 30-node regional medical consortium chain, the bandwidth savings for the leader and the corresponding cloud costs can be conservatively estimated based on the data in Section 4. First, it can be known from the peak throughput that Raft achieves 23 800 transactions/s and MH-Raft achieves 32 000 transactions/s. Based on the common transaction size of 250 bytes per transaction in consortium chains, the Raft leader needs to broadcast directly to all 29 followers, and the uplink peak bandwidth is $23\,800 \times 250 = 5.95$ MB/s. In contrast, MH-Raft sends messages to only approximately 8.26 mediators through the tightness mediator mechanism, reducing redundant information and simplifying the node handshake mechanism. The corresponding theoretical bandwidth reduction ratio is approximately $(29 - 8.26)/29 \times 100\% \approx 71.5\%$. Accordingly, the peak uplink bandwidth of the MH-Raft leader is reduced to $5.95\ \text{MB/s} \times (1 - 0.715) \approx 1.70$ MB/s, saving 4.25 MB/s.

If conservatively estimated based on 24-hour continuous peak operation, the daily traffic difference is $4.25\ \text{MB/s} \times 86\,400\ \text{s} \approx 367$ GB, and the monthly traffic difference is approximately 11 TB. Based on the public cloud traffic billing rate of 0.8532 CNY/GB, leader nodes can save approximately 9491.85 CNY in cloud costs each month. If a single logistics node goes offline, the remaining nodes in this subset group can still meet the confirmation requirements of more than half of the nodes, and the mediator's return of the ACK will not be hindered. If there is a 200 ms latency across regions, since the consensus latency of MH-Raft has been reduced to 168.7 ms, it can still be controlled at 169 ms + 200 ms = 369 ms after amplification, and leader re-election will not be triggered. The

above analysis is based entirely on the publicly available data in the study, proving that MH-Raft can still achieve practical value, such as sub-second confirmation, low cost, and easy operation and maintenance in cross-regional, high-concurrency, and strongly supervised regional medical scenarios.

6 Conclusions

In this study, we present an improved Raft consensus algorithm called MH-Raft. First, we address the selection of a qualified leader as an optimization problem, directly determining the leader node using the MOEA/D algorithm. Then, we select follower nodes that are more suitable to serve as mediator nodes by improving the definition of tightness values. Concurrently, we design a hybrid clustering algorithm to determine the number of mediator nodes and perform clustering through an adaptive clustering algorithm. Finally, we improve the matching method between mediator nodes and other follower groups. Experimental results show that the election latency is significantly reduced, the consensus latency is reduced by 14.87%–34.45% from small-scale to large-scale nodes, and the throughput is increased by an average of 30.43%. This approach effectively bridges the gap between topological proximity and node fitness, providing a scalable and resilient solution for large-scale distributed consensus. These findings indicate that the MH-Raft algorithm has potential advantages in real-world scenarios. We believe that such adaptive and secure mechanisms will be instrumental in the evolution of next-generation blockchain networks capable of handling complex industrial workloads.

This study demonstrates that integrating intelligent optimization strategies with a structural layer is a viable paradigm for overcoming the scalability bottlenecks of distributed consensus protocols. To address remaining challenges in ultra-large-scale deployments, future work will focus on minimizing structural maintenance and mediator-matching overheads, exploring lightweight heuristic algorithms to accelerate convergence, and enhancing the system's resilience against Byzantine faults.

Acknowledgments

This work was supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM413), the Flexible Introduction of Leading Talents under the 2023 Kunlun Talents High-End Innovation and Entrepreneurship Talents Project of Qinghai Province (No. QHKLYC-GDCXCY-2023-320), the Qinghai University Research Ability Enhancement Project (No. 2025KTSA01), the "Unveiling the Leader" Science and Technology R&D Projects (No. 2022ZXJ03C06), and the National Natural Science Foundation of China (No. 62076082).

Author contributions

Fei ZHAO designed the research, processed the data, analyzed the results, and drafted the paper. Guilong PENG constructed the theoretical framework. Tianyi ZANG managed the project and helped organize the paper. All the authors revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration on the use of generative AI tools

During the preparation of this work, the authors used ERNIE to improve language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Abbasihafshejani M, Manshaei MH, Jadliwala M, 2023. Detecting and punishing selfish behavior during gossiping in algorand blockchain. IEEE Virtual Conf on Communications, p.49-55. https://doi.org/10.1109/VCC60689.2023.10474784

Attaran M, 2022. Blockchain technology in healthcare: challenges and opportunities. *Int J Healthc Manage*, 15(1):70-83. https://doi.org/10.1080/20479700.2020.1843887

Bagaria V, Kannan S, Tse D, et al., 2019. Prism: deconstructing the blockchain to approach physical limits. Proc ACM SIGSAC Conf on Computer and Communications Security, p.585-602. https://doi.org/10.1145/3319535.3363213

Ben Othmen R, Abbessi W, Ouni S, et al., 2023. Simulation of optimized cluster based PBFT blockchain validation process. IEEE Symp on Computers and Communications, p.1317-1322. https://doi.org/10.1109/iscc58397.2023.10218119

Bentov I, Lee C, Mizrahi A, et al., 2014. Proof of activity: extending Bitcoin's proof of work via proof of stake. *ACM SIGMETRICS Perform Eval Rev*, 42(3):34-37. https://doi.org/10.1145/2695533.2695545

Castro M, Liskov B, 1999. Practical Byzantine fault tolerance. Proc 3rd Symp on Operating Systems Design and Implementation, p.173-186. https://dl.acm.org/doi/10.5555/296806.296824

Chatterjee D, Banerjee P, Mazumdar S, 2023. Chrisimos: a useful proof-of-work for finding minimal dominating set of a graph. IEEE 22nd Int Conf on Trust, Security and Privacy in Computing and Communications, p.1332-1339. https://doi.org/10.1109/TrustCom60117.2023.00182

Chen L, Xu L, Shah N, et al., 2017. On security analysis of proof-of-elapsed-time (PoET). Proc 19th Int Symp on Stabilization, Safety, and Security of Distributed Systems, p.282-297. https://doi.org/10.1007/978-3-319-69084-1_19

Chen YF, Guo Y, Wang MY, et al., 2024. Securing IOTA blockchain against tangle vulnerability by using large deviation theory. *IEEE Int Things J*, 11(2):1952-1965. https://doi.org/10.1109/JIOT.2023.3283788

Cheng JX, Chen YZ, Cao YZ, et al., 2024. A vulnerability detection framework by focusing on critical execution paths. *Inform Softw Technol*, 174:107517. https://doi.org/10.1016/j.infsof.2024.107517

Coelho IM, Coelho VN, Araujo RP, et al., 2020. Challenges of PBFT-inspired consensus for blockchain and enhancements over Neo dBFT. *Fut Int*, 12(8):129. https://doi.org/10.3390/fi12080129

Eichelberger H, Sauer C, Ahmadian AS, et al., 2025. Industry 4.0/IIoT platforms for manufacturing systems-a systematic review contrasting the scientific and the industrial side. *Inform Softw Technol*, 179:107650. https://doi.org/10.1016/j.infsof.2024.107650

Escobar CC, Roy S, Kreidl OP, et al., 2022. Toward a green blockchain: engineering Merkle tree and proof of work for energy optimization. *IEEE Trans Netw Serv Manag*, 19(4):3847-3857. https://doi.org/10.1109/TNSM.2022.3219494

Eyal I, Gencer AE, Sirer EG, et al., 2016. Bitcoin-NG: a scalable blockchain protocol. 13th USENIX Symp on Networked Systems Design and Implementation, p.45-59.

Gadiraju DS, Lalitha V, Aggarwal V, 2023. An optimization framework based on deep reinforcement learning approaches for Prism blockchain. *IEEE Trans Serv Comput*, 16(4):2451-2461. https://doi.org/10.1109/TSC.2023.3242606

Gilad Y, Hemo R, Micali S, et al., 2017. Algorand: scaling Byzantine agreements for cryptocurrencies. Proc 26th Symp on Operating Systems Principles, p.51-68. https://doi.org/10.1145/3132747.3132757

Hu TY, Li BX, 2025. Dynamic information utilization for securing Ethereum smart contracts: a literature review. *Inform Softw Technol*, 182:107719. https://doi.org/10.1016/j.infsof.2025.107719

Jakobsson M, Juels A, 1999. Proofs of work and bread pudding protocols. In: Preneel, B (Eds.), Secure Information Networks. Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conf on Communications and Multimedia Security, volume 23. Springer, Boston, p.258-272. https://doi.org/10.1007/978-0-387-35568-9_18

Ji SY, Wu J, Qiu JF, et al., 2023. Effuzz: efficient fuzzing by directed search for smart contracts. *Inform Softw Technol*, 159:107213. https://doi.org/10.1016/j.infsof.2023.107213

Jia B, Zhou T, Li W, et al., 2018. A blockchain-based location privacy protection incentive mechanism in crowd sensing networks. *Sensors*, 18(11):3894. https://doi.org/10.3390/s18113894

Jing N, Liu Q, Sugumaran V, 2021. A blockchain-based code copyright management system. *Inform Process Manag*, 58(3):102518. https://doi.org/10.1016/j.ipm.2021.102518

Keidar I, Kokoris-Kogias E, Naor O, et al., 2021. All you need is DAG. Proc ACM Symp on Principles of Distributed Computing, p.165-175. https://doi.org/10.1145/3465084.3467905

Kovalchuk L, Oliynykov R, Bespalov Y, et al., 2022. Comparative analysis of consensus algorithms using a directed acyclic graph instead of a blockchain, and the construction of security estimates of spectre protocol against double spend attack. In: Oliynykov R, Kuznetsov O, Lemeshko O, et al. (Eds.), Information Security Technologies in the Decentralized Distributed Networks. Lecture Notes on Data Engineering and Communications Technologies, volume 115. Springer, Cham, p.203-224. https://doi.org/10.1007/978-3-030-95161-0_9

Laatikainen G, Li MC, Abrahamsson P, 2023. A system-based view of blockchain governance. *Inform Softw Technol*, 157:107149. https://doi.org/10.1016/j.infsof.2023.107149

Lamport L, 2019. The part-time parliament. *ACM Trans Comput Syst*, 16(2):133-169. https://doi.org/10.1145/279227.279229

Lasla N, Al-Sahan L, Abdallah M, et al., 2022. Green-PoW: an energy-efficient blockchain proof-of-work consensus algorithm. *Comput Netw*, 214:109118. https://doi.org/10.1016/j.comnet.2022.109118

Liu X, Huang Z, Wang Q, et al., 2022. An optimized key-value Raft algorithm for satisfying linearizable consistency. Int Conf on Networking and Network Applications, p.522-527. https://doi.org/10.1109/NaNA56854.2022.00096

Liu YR, Shi TJ, 2023. Improved A-Raft consensus algorithm based on SHA256 encryption algorithm. Int Conf on the Cognitive Computing and Complex Data, p.317-322. https://doi.org/10.1109/ICCD59681.2023.10420609

Ongaro D, Ousterhout J, 2014. In search of an understandable consensus algorithm. Proc USENIX Conf on USENIX Annual Technical Conf, p.305-320. https://dl.acm.org/doi/10.5555/2643634.2643666

Shi CC, Xiang Y, Yu JS, et al., 2023. Machine translation-based fine-grained comments generation for solidity smart contracts. *Inform Softw Technol*, 153:107065. https://doi.org/10.1016/j.infsof.2022.107065

Shi HR, Chen ZH, Cheng YQ, et al., 2025. PB-Raft: a Byzantine fault tolerance consensus algorithm based on weighted PageRank and BLS threshold signature. *Peer-to-Peer Netw Appl*, 18(1):26. https://doi.org/10.1007/s12083-024-01876-8

Sokal RR, Michener CD, 1958. A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull*, 38(22):1409-1438.

Sun WY, Bai XM, Shi BS, et al., 2025. TD-Raft: a consensus algorithm with inspection mechanism and server performance threshold. *IAENG Int J Comput Sci*, 52(4):1070-1076.

Tang H, Yi WL, Zhao YD, et al., 2022. Improved Raft algorithm for optimizing authorized nodes based on random forest. XXV Int Conf on Soft Computing and Measurements, p.279-282. https://doi.org/10.1109/SCM55405.2022.9794853

Treleaven P, Brown RG, Yang D, 2017. Blockchain technology in finance. *Computer*, 50(9):14-17. https://doi.org/10.1109/MC.2017.3571047

Ulukök MK, Sariyildiz İ, Evrim V, 2025. Hybrid Raft-PoW blockchain consensus algorithm. *IEEE Access*, 13:72067-72076. https://doi.org/10.1109/ACCESS.2025.3562725

Wang Q, Yu JS, Peng ZN, et al., 2020. Security analysis on dBFT protocol of Neo. 24th Int Conf on Financial Cryptography and Data Security, p.20-31. https://doi.org/10.1007/978-3-030-51280-4_2

Ward JH, 1963. Hierarchical grouping to optimize an objective function. *J Am Stat Assoc*, 58(301):236-244. https://doi.org/10.1080/01621459.1963.10500845

Yamashita A, Tanaka M, Bessho Y, et al., 2023. Improving Raft performance with bulk transfers. 11th Int Symp on Computing and Networking Workshops, p.38-44. https://doi.org/10.1109/CANDARW60564.2023.00015

Yang F, Zhou W, Wu QQ, et al., 2019. Delegated proof of stake with downgrade: a secure and efficient blockchain consensus algorithm with downgrade mechanism. *IEEE Access*, 7:118541-118555. https://doi.org/10.1109/ACCESS.2019.2935149

Yang SJ, Tan PL, Fu HW, 2024. Improved Raft consensus algorithm based on NSGA-II and K-means++. 10th Int Symp on System Security, Safety, and Reliability, p.383-390. https://doi.org/10.1109/isssr61934.2024.00055

Yin MF, Malkhi D, Reiter MK, et al., 2019. HotStuff: BFT consensus with linearity and responsiveness. Proc ACM Symp on Principles of Distributed Computing, p.347-356. https://doi.org/10.1145/3293611.3331591

Zhai D, Wang J, Liu JQ, et al., 2023. Efficient-HotStuff: a BFT blockchain consensus with higher efficiency and stronger robustness. 28th Int Conf on Parallel and Distributed Systems, p.217-225. https://doi.org/10.1109/ICPADS56603.2022.00036

Zhang QF, Li H, 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans Evol Comput*, 11(6):712-731. https://doi.org/10.1109/TEVC.2007.892759

Zhao JJ, Chen X, Yang G, et al., 2024. Automatic smart contract comment generation via large language models and in-context learning. *Inform Softw Technol*, 168:107405. https://doi.org/10.1016/j.infsof.2024.107405

Zhao WB, 2023. On Nxt proof of stake algorithm: a simulation study. *IEEE Trans Depend Sec Comput*, 20(4):3546-3557. https://doi.org/10.1109/tdsc.2022.3193092