An Efficient Blockchain Neighbor Selection Framework Based on Agglomerative Clustering

Marwa F. Mohamed, Mostafa Elkhouly*, Safa Abd El-Aziz, Mohamed Tahoun
Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University,
Ismailia, Egypt 41522

***Abstract*—Blockchain-based decentralized applications have garnered significant attention and have been widely deployed in recent years. However, blockchain technology faces several challenges, such as limited transaction throughput, large blockchain sizes, scalability, and consensus protocol limitations. This paper introduces an efficient framework to accelerate broadcast efficiency and enhance the blockchain system's throughput by reducing block propagation time. It addresses these concerns by proposing a dynamic and optimized Blockchain Neighbor Selection Framework (BNSF) based on agglomerative clustering. The main idea behind the BNSF is to divide the network into clusters and select a leader node for each cluster. Each leader node resolves the Minimum Spanning Tree (MST) problem for its cluster in parallel. Once these individual MSTs are connected, they form a comprehensive MST for the entire network, where nodes obtain optimal neighbors to facilitate the process of block propagation. The evaluation of BNSF showed superior performance compared to neighbor selection solutions such as Dynamic Optimized Neighbor Selection Algorithm (DONS), Random Neighbor Selection (RNS), and Neighbor Selection based on Round Trip Time (RTT-NS). Furthermore, BNSF significantly reduced the block propagation time, surpassing DONS, RTT-NS, and RNS by 51.14%, 99.16%, and 99.95%, respectively. The BNSF framework also achieved an average MST calculation time of 27.92% lower than the DONS algorithm.**

***Keywords*—Blockchain; scalability; agglomerative clustering; broadcasting; optimized neighbor selection; minimum spanning tree; parallel processing**

I. Introduction

Blockchain (BC) is a decentralized ledger technology that operates on a peer-to-peer (P2P) network, utilizing a cryptographic chain of blocks and consensus algorithms to verify and store data in decentralized networks [1]. BC was initially introduced in 2008, credited to Satoshi Nakamoto [2]. It enables nodes that do not have mutual trust to reach a consensus on a sequential collection of blocks containing multiple transactions, all without the need for a third party [3]. In recent years, BC has garnered increasing attention due to its numerous advantages compared to traditional databases [4]. BC is immutable, transparent, secure, and decentralized, resulting in a significant reduction in the likelihood of a Single Point of Failure (SPF) [5]. This enhances its reliability and efficiency in comparison to conventional data storage systems. The networks within BC can manage information securely and protect it from tampering, even when there are many malicious nodes [6]. In addition, no third-party authentication is required, as BC operates without central management. These features are highly valuable and find application not only in cryptocurrencies but also in a wide range of fields [7]. Therefore, BC has a broad spectrum of applications in emerging fields such as 5G [8], [9], [10], smart cities [11], [12], [13], the internet of things [14], [15], [16], social networking [17], [18], [19], and artificial intelligence [20], [21], [22].

Although BC has many great advantages, it still has some drawbacks, such as the scalability problem that arises when the number of users in the BC system increases significantly. Scalability in BC is typically measured in transactions per second (TPS) [23], [24]. A more scalable BC allows for a higher number of transactions between network nodes, resulting in increased bandwidth consumption and network latency. Consequently, the primary challenge with BC technology lies in its low transaction transfer rate and approval time. For instance, Bitcoin can handle only 7 TPS, resulting in significantly lower throughput compared to widely used mainstream payment platforms such as PayPal, which achieves a transfer rate of 500 TPS, and Visa, which surpasses 4000 TPS. Ethereum is Another example that can achieve approximately 15 TPS [25]. Obviously, neither Bitcoin nor Ethereum can meet the demands of large-scale trading scenarios.

BC is mainly composed of three layers: the data layer, the consensus layer, and the network layer [26]. Within the data layer, there exists a chain of interconnected data blocks, supported by hashing algorithms and Merkle trees to protect the integrity and traceability of block data. The consensus layer encompasses a variety of consensus algorithms that facilitate data consistency among network nodes [27]. On the other hand, the network layer comprises mechanisms for propagating data and verifying transactions [28], [29].

Solutions for BC scalability are classified by implementation layer [30]. State-of-the-art BC research addresses scalability in three key areas. In the data layer, compression reduces transaction and block sizes, minimizing bandwidth use [31]. The consensus layer improves communication for faster transactions and lower latency [32]. In the network layer, the gossip algorithm and P2P structure are optimized for enhanced peer communication, boosting BC system performance [33], [34].

Gossip broadcasting in the BC network results in the duplication of information and inefficient bandwidth utilization. However, as the number of peers joining the network increases, duplication and bandwidth utilization also increase due to a higher probability of selected peers interfering with the gossip process [35]. Therefore, alternative techniques for broadcasting blocks in the network, such as Random Neighbor Selection (RNS), where shared data propagates through random paths [36], lead to an inefficient data propagation scheme. This inefficiency arises from the probability of redundancy in the exchanged messages between network nodes. This redundancy

occurs due to cycling in the randomly chosen data paths resulting in longer delivery times and lower levels of consistency. Nevertheless, most BC systems support RNS. Some methods have been proposed to improve the Neighbor Selection (NS) process locally, addressing the dynamicity problem. Bi et al. [37] introduced an NS protocol based on network latency, where nodes assess the Round Trip Time (RTT) to their neighboring nodes. Consequently, nodes prioritize neighbors with the lowest RTT for the NS process. Nonetheless, none of these solutions has proposed an ideal NS strategy.

In this paper, an Efficient Blockchain Neighbor Selection Framework (BNSF) is introduced to accelerate block propagation and enable node communication with selected neighbors. The network is divided into clusters using agglomerative clustering. Within each cluster, a leader node is chosen to resolve the Minimum Spanning Tree (MST) problem using Dijkstra's Algorithm. Subsequently, the MST for the entire network is obtained by connecting the MSTs from the network clusters.

The key contributions of this paper are summarized as follows:

1) Addressing the scalability issue of the BC network by optimizing the NS process in a dynamic network topology.
2) Reducing the total calculation time to construct the general MST for the entire network by dividing the network into clusters using agglomerative clustering, constructing the MST for each cluster, and finally connecting them to obtain the general MST.
3) Utilizing multi-threading technology: each cluster computes the MST in parallel to accelerate execution time. This approach also takes advantage of multiple CPUs or cores, resulting in further performance improvements.
4) Reducing duplicates in data exchanged between network nodes, as each node shares data with its MST optimal neighbors (MON) without cycling in selected paths.
5) Reducing the total propagation time of exchanged data between network nodes.

The remaining sections of this paper are structured as follows: Section II analyzes relevant literature, Section III provides a detailed explanation of the proposed BNSF, Section IV presents the evaluation of BNSF, and finally, Section V summarizes the most significant findings and conclusions.

II. Related Work

In this section, several modern network layer scalability solutions are presented. These solutions primarily focus on enhancing either the gossip algorithm or the P2P network architecture. Research studies aiming to improve the gossip algorithm focus on reducing duplicate data or increasing block propagation speed [38]. The proposed solutions aim to decrease the level of duplication caused by the gossip algorithm or to reduce block propagation time through an enhanced gossip protocol. Following are some of the recent work representing such solutions.

The Fastchain protocol, designed to enhance the scalability of BC as described in [39], operates through a mechanism in which a node with limited bandwidth transmits a block to a node possessing higher bandwidth capacity. Subsequently, the latter node distributes the block to all other nodes in the network. Nodes with restricted bandwidth prioritize connections with nodes that possess higher bandwidth and disconnect from nodes whose bandwidth is less than a specific threshold. The implementation of Fastchain comprises two essential stages, namely the bandwidth monitoring phase and the neighbor update phase. In the bandwidth monitoring phase, every node maintains a table containing the recent bandwidth information of its neighboring nodes. During the neighbor update phase, nodes periodically update their connections with neighbors, continuously disconnecting from those with slow and low bandwidth. FastChain enhances the effective block rate, resulting in a 40% increase in the number of blocks added to the chain compared to bitcoin. Furthermore, it improves throughput by 20% to 40%.

Baniata and Anaqreh [40] introduced a Dynamic Optimized Neighbor Selection Algorithm (DONS) for P2P network management within the BC. A leader peer is selected to oversee the network and construct its topology using neighbor lists from regular peers. The resulting MST guides the leader in identifying optimal neighbors, enhancing transaction throughput by minimizing propagation delay. However, leader changes necessitate network topology reconstruction and requesting neighbors' lists. With growing peer numbers, MST computation time increases, leading to inefficient bandwidth use. Additionally, leader unavailability risks both topology loss and reselection overheads.

BlockP2P [41] is a clustering method designed to enhance transaction throughput by reducing the latency within the BC network. It proposes to group BC nodes into clusters based on their geographic location, which leads to a cluster with a small diameter and high connectivity, thus reducing the diffusion time within the block. The authors defined three types of nodes, leaf nodes, core nodes, and a routing node for each cluster, which is randomly selected from the core nodes. Routing nodes in different clusters are interconnected to forward transactions or blocks, thus ensuring full connectivity between clusters. Transaction throughput increased by about 90% due to reduced latency. The clustering method has better bandwidth efficiency with a small network size compared to random neighbor selection. However, congestion can occur in the cluster as the network grows and the efficiency within the cluster decreases. This approach is susceptible to network partitioning and over-reliance on a single node.

The authors in [42], [43] proposed a score-based NS protocol for constructing a BC network. This protocol assigns higher scores to peers with lower propagation delays compared to peers with higher propagation delays. Subsequently, peers with the highest scores are chosen as neighbors. Every miner node assesses its neighboring nodes based on the disparity between the time the block was created and the time it was received at the recipient node. Once a node successfully receives ten blocks, it proceeds to update its list of neighbors. In this update, the node randomly selects new neighbors and includes only those with high scores. Neighbor nodes exhibiting faster transfer of new blocks compared to other neighbors are assigned higher scores, indicating superior network communications capabilities. Thus, miners prefer neighbors

with higher scores in the NS process. This method leads to excessive dependence on the nodes that have the shortest total propagation time, which can reduce node performance.

Deshpande et al. [44] proposed a centralized solution. This solution utilizes the principles of Software-Defined Networking (SDN) to reduce the excessive overhead in managing a distributed network for blockchains. Servers create a P2P topology using clustering techniques and assign neighbors to each peer using the RNS method. Unlike other clustering-based approaches, the proposed method offered a flexible means of network management, incorporating constraints to mitigate congestion issues within the cluster. In the proposed centralized network model, topology management has demonstrated a notable reduction in bandwidth consumption compared to the traffic caused by managing distributed network models. This approach can improve the transfer rate of transactions in BC networks. Due to reduced responsibilities, network peers can allocate all available resources to process a greater number of transactions. However, it should be noted that as the network size grows, the time required for calculating the structure also increases.

Vu and Tewari [45] proposed a probability-based gossiping method for neighbor selection. A network node sends several inventory messages (INV) that are used in Bitcoin and count the number of responses received. The sending and receiving ratio is the probability used to determine which neighbor gets the new block. As a result of this approach, there was a reduction in the number of messages transmitted by the network nodes. Additionally, this approach reduces duplication compared to the default gossip protocol employed in Bitcoin. Moreover, probability calculations are not disregarded but retained for subsequent transmissions, as well as the size of the network. However, excessive and frequent sending of INV messages between network nodes results in network overhead and consumption of network resources.

The authors in [46] propose Trust-based Optimum Neighbor Selection (TONS), an optimized algorithm for blockchain networks in IoT environments, addressing the challenge of unreliable or malicious nodes. TONS employs a trust and reputation model to evaluate node reliability, ensuring miners communicate with the most trustworthy neighbors. The algorithm computes optimal neighbor selection considering both delivery time rates and node reputation. Experimental simulations show TONS outperforms traditional methods in efficiency and effectiveness. However, TONS introduces a high time cost for computing trust measures, and the energy consumption associated with computing trust measures between nodes increases.

Table I summarizes the main works that have addressed the neighbor selection problem in BC networks.

III. Blockchain Neighbor Selection Framework (BNSF)

In this section, a detailed explanation of the proposed BNSF is provided, including all the used methods and implemented algorithms as well. The proposed framework analyzes and evaluates an alternative method for selecting neighbors for the Gossip communication protocol in a public BC network to accelerate the final latency. Furthermore, it introduces a multi-leader scenario to reduce the calculation time of the MST topology for the entire network as the network size increases. Fig. 1 illustrates the BNSF architecture.

A. The Proposed System Model

The examined permission-less public BC network topology denoted as G, consists of a set of nodes $S = \{s_1, s_2, \ldots, s_N\}$, where N represents the total number of nodes within the network. The set S is divided into a set of clusters $\mathbf{C} = \{c_1, c_2, \ldots, c_M\}$, where $M \leq N$. Each cluster $c_i \in \mathbf{C}$ comprises a set of nodes $S_i = \{s_1, s_2, \ldots, s_{n_i}\}$, with $i = 1, 2, 3, \ldots, M$. The value of N is calculated as follows:

$$N = \sum_{i=1}^{M} n_i \tag{1}$$

Each cluster $c_i \in \mathbf{C}$ can be represented as a weighted undirected graph $G_i = (S_i, E_i, W_i)$. S_i denotes the set of nodes in cluster c_i, $E_i = \left\{e_{s_i s_j} \mid s_i, s_j \in S_i\right\}$ represents the finite set of edges (i.e., communication channels) connecting the nodes, and $W_i = \left\{w_{e_{s_i s_j}} \mid e_{s_i s_j} \in E_i\right\}$ is a finite set of weights assigned to E_i. It can be represented as a function $W_i : E_i \to \mathbb{R}^+$, where $\mathbb{R}^+$ denotes the set of positive real numbers.

The MST for cluster c_i in G_i is denoted as $MST_i = (S_i, T_i, W_i^{MST})$, where S_i represents the set of nodes, T_i denotes the set of edges forming the MST, and W_i^{MST} is a finite set of weights assigned to T_i. Similarly to before, the weights are defined by the function $W_i^{MST} : T_i \to \mathbb{R}^+$.

Each node $s_j \in S_i$, where $j = 1, 2, 3, \ldots, n_i$, has a neighbor set denoted by $\mathscr{N}_{c_i}(s_j)$. The neighbor set $\mathscr{N}_{c_i}(s_j)$ consists of nodes that are directly connected to s_j within the cluster c_i. This can be represented as:

$$\mathscr{N}_{c_i}(s_j) = \{s_k \mid s_k \in S_i, s_k \neq s_j, (s_j, s_k) \in E_i\} \tag{2}$$

E_i represents the set of edges in the graph G_i associated with cluster c_i. The expression $(s_j, s_k) \in E_i$ checks if there exists an edge between nodes s_j and s_k in the graph G_i associated with cluster c_i. The condition $s_k \neq s_j$ ensures that s_j is not included in its own neighbor set. With this notation, each node $s_j \in S_i$ is aware of its neighbor set $\mathscr{N}_{c_i}(s_j)$.

The edge matrix A_E is an $N \times N$ matrix with elements $\{e_{s_i s_j}\}$, where $i, j = 1, 2, 3, \ldots, N$. It represents the connectivity and relationships between nodes in the network G. Each element $e_{s_i s_j}$ in the matrix represents the presence or absence of an edge between nodes s_i and s_j.

The distance between clusters c_i and c_j is represented as $D(c_i, c_j)$. It is initialized with the distances between nodes $s_i \in c_i$ and $s_j \in c_j$, where $s_i, s_j \in \{(i, j) | i = 1, 2, \ldots, n_i, j = 1, 2, \ldots, n_j\}$. The distance between clusters c_i and c_j is determined using the Complete-linkage method, which selects the largest distance among all pairs of nodes $s_i \in c_i$ and $s_j \in c_j$:

$$D(c_i, c_j) = \max_{s_i \in c_i, s_j \in c_j} \{D(s_i, s_j)\} \tag{3}$$

The distance between nodes s_i and s_j is calculated using the Euclidean distance formula:

TABLE I. A COMPARISON OF THE PROPOSED FRAMEWORK WITH RELATED WORK. NOTABLE ABBREVIATIONS: PB - PUBLIC BC, DT - DYNAMIC NETWORK TOPOLOGY, CL - CLUSTERING, GV - GLOBAL VIEW, LN - EFFECTIVE IN LARGE NETWORKS

Ref	PB	DT	Cl	GV	LN	Limitations
[39]	✓	✓	×	×	×	Each node must maintain the latest bandwidth table which periodically updates neighbor connections to get the latest update. Nodes with limited bandwidth always rely on the highest bandwidth nodes
[40]	✓	✓	×	✓	×	The network topology calculation time increases with the size of the network. The overhead incurred by frequent leader selections
[41]	×	✓	✓	×	×	The network is vulnerable to congestion and over-reliance on a single node in network traffic
[42], [43]	×	×	×	×	×	Network excessively depends on a single node with the shortest propagation time. Consequently, it is prone to congestion.
[44]	✓	✓	✓	×	×	As the number of nodes increases, the calculation time for network topology also rises.
[45]	✓	×	×	×	×	The excessive and frequent transmission of INV messages leads to network overhead.
[46]	✓	×	×	✓	✓	High time cost for computing trust measures and the increased energy consumption.
BNSF	✓	✓	✓	✓	✓	

$$D(s_i, s_j) = \sqrt{(s_i.x - s_j.x)^2 + (s_i.y - s_j.y)^2} \quad (4)$$

The collection of root nodes of the MST for all clusters can be denoted as:

$$R = \bigcup_{i=1}^{M} r_i \quad (5)$$

Here, r_i denotes the root node of its corresponding cluster c_i. The union symbol $\bigcup$ indicates the combination of root nodes from all clusters, forming the collection R. Subsequently, the proposed framework establishes connections among all these root nodes, creating a comprehensive MST for the entire BC Network.

Optimal neighbor nodes for a given node s_i can be represented as $MON\,(s_i) = \{(s_1, w_1), (s_2, w_2), \ldots, (s_n, w_n)\}$, where each pair $((s_j, w_j))$ denotes an optimal neighbor node s_j and its corresponding weight value w_j for the node s_i.

The MST_{c_i} of each cluster c_i is computed in a separate thread to reduce BNSF processing time, which is represented as $x_i \in \mathcal{X}$. The set of threads $\mathcal{X}$, denoted as $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, encompasses all the threads involved in calculating the MSTs of the clusters. Each element $x_i \in \mathcal{X}$ represents an individual thread responsible for computing the MST_{c_i} of cluster c_i. Table II summarizes the main symbols used in the BNSF model.

In the following sections, the phases of the proposed BNSF framework are explained in detail.

B. Phase 1: Network Clustering

Agglomerative Clustering (AC) is applied in a bottom-up manner to group network nodes by considering their similarities [47]. Initially, each node is treated as an individual cluster. Subsequently, clusters are successively combined until all nodes are contained within a single large cluster. At each iteration of the algorithm, the two clusters c_i and c_j that have not been previously merged are examined, and the distance D between the two clusters is computed. The pair with the minimum value in distance D is then selected and joined to form a new cluster, denoted as c_{new}. Once the clusters are joined, the algorithm proceeds to calculate the distances $D(c_{\text{new}}, c_k)$ between the newly formed cluster c_{new} and all other clusters. This operation is repeated until the cluster set $\mathbf{C}$ with size M is constructed (Fig. 2(B)).

TABLE II. LIST OF SYMBOLS USED IN THE BNSF MODEL

S	Set of nodes within the network G.
N	Total number of nodes.
n_i	Number of nodes within cluster c_i, where n_i is a subset of N.
$\mathbf{C}$	Set of clusters within the network G.
c_i	Cluster of nodes, where $c_i \in \mathbf{C}$.
M	Number of clusters within the network G.
S_i	Set of nodes within cluster c_i, where $i \leq M$.
s_j	Network node, where $s_j \in S_i$.
$\mathcal{N}_{c_i}(s_j)$	Neighbor set for every node $s_j \in S_i$ in cluster c_i.
k	Number of neighbors for every node $s_j \in S_i$.
E_i	Set of edges within the network G_i of cluster c_i.
W_i	Set of weights within the network G_i of cluster c_i.
A_E	Edge matrix.
$D(s_i, s_j)$	Distance between two nodes $s_i \in c_i$ and $s_j \in c_j$.
$MON\,(s_i)$	Set of optimal neighbor nodes s_j for the node s_i.
$\mathcal{X}$	List of n threads, where each $x_i \in \mathcal{X}$ represents an individual thread.

In **Step-1**, the BNSF framework applies AC Algorithm 1 as follows:

First, the network graph G is converted into an edge matrix A_E for AC application. Then, distance or similarity information is calculated for every pair of nodes using Eq. 4. Next, the complete linkage function is employed to group the nodes into a hierarchical cluster tree. Close clusters are linked to each other using the linkage function. Complete-linkage clustering, also known as farthest-neighbor aggregation [48], is a method of AC for calculating the distance between clusters in hierarchical clustering, as shown in Eq. 3.

In complete linkage, the distance $D(c_i, c_j)$ between two clusters c_i and c_j is determined as the maximum distance observed between any individual node s_i in the first cluster c_i and any individual node s_j in the second cluster c_j. The dissimilarity between clusters c_i and c_j is defined as $max\ D\,(s_i, s_j)$, where $s_i \in c_i$ and $s_j \in c_j$. The two clusters c_i and c_j that exhibit the highest similarity with the minimum value in D are merged into a new cluster, denoted as $c_{\text{new}} = c_i \cup c_j$.

Afterward, determine the point at which to divide the hierarchical tree into clusters by specifying the number of clusters M. Then, apply AC to the network edge matrix A_E until the desired number of clusters M is achieved. Finally,

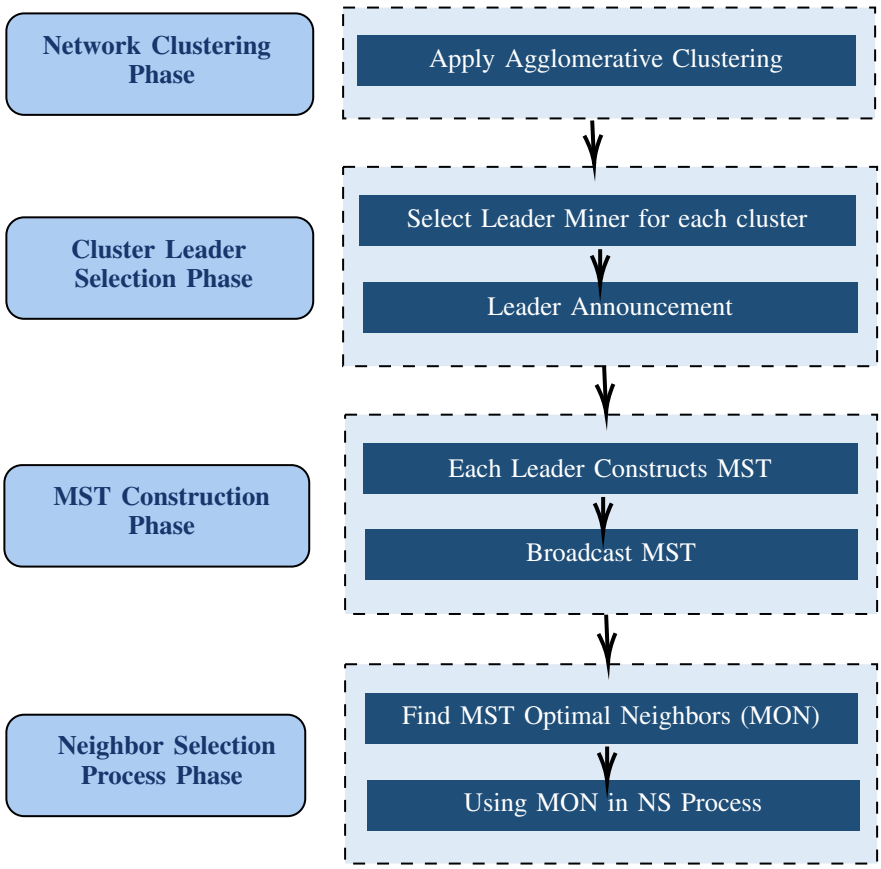

Fig. 1. The main steps involved in the proposed BNSF framework.

the cluster set $\mathbf{C}$ is obtained through the application of AC.

C. Phase 2: Cluster Leader Selection

This phase is responsible for two main steps: cluster leader selection and leader announcement. The BNSF framework requires a global view of the BC network. All nodes $s_j \in S$ have equal privileges in the public and permissionless BC network G. However, the proposed BNSF selects one of these nodes to perform MST calculations for all other nodes. Each cluster of nodes c_i needs to choose a single node $s_i \in S_i$ as its Leader Node (LN). LN possesses more privileges than other nodes in the same cluster, granting it a global view of the entire cluster. Additionally, LN collects information from the other nodes within the same cluster and uses it to generate the MST for the entire cluster c_i. Thus, each node s_i in cluster c_i can select its optimal neighbors from the generated MST for exchanging new blocks or transactions. Moreover, the network's global view is influenced by nodes joining or leaving, necessitating regular updates to the calculated MST to accommodate changes in the network G.

In **Step-2**, the cluster leader selection proposed by the BNSF framework can be described as follows:

A random leader selection scenario is proposed. For each cluster c_i in the network topology G, BNSF selects a cluster node $s_i \in S_i$ to be the LN of its cluster c_i. The LN is randomly chosen to build the MST for its cluster c_i. Random leader selection enhances network security because attackers cannot

Algorithm 1 Apply Agglomerative Clustering

Input: Number of nodes N, Number of clusters M and Set of nodes S
Output: Clusters set $\mathbf{C}$.

Set the edge matrix $A_E = 0$. /* Initialize A_E */
for $j \leftarrow 1$ to N **do**
  for $k \leftarrow 1$ to N **do**
    if s_k is a neighbor of s_j **then**
      set $e_{s_j s_k} \leftarrow 1$ within the edge matrix A_E.
    end if
  end for
end for
  /* Apply Agglomerative Clustering(M, S, N) on A_E*/
$\mathbf{C} = \{c_1, c_2, \ldots, c_N\}$, where each c_i contains one node s_i. /* Initialize $\mathbf{C}$*/
Calculate $D(c_i, c_j)$ between every pair of clusters $c_i, c_j \in \mathbf{C}$ using E.q. 3
while $M < length(\mathbf{C})$ **do** /*where M is the desired number of $\mathbf{C}$ */
  Find the pair of clusters with minimum $D(c_i, c_j)$
  $c_{new} \leftarrow c_i \cup c_j$.
  Remove c_i, c_j from $\mathbf{C}$ and add c_{new} to $\mathbf{C}$ /* Update $\mathbf{C}$. */
  Calculate $D(c_{new}, c_k)$, where c_k represents the other clusters.
end while
Return $\mathbf{C}$.

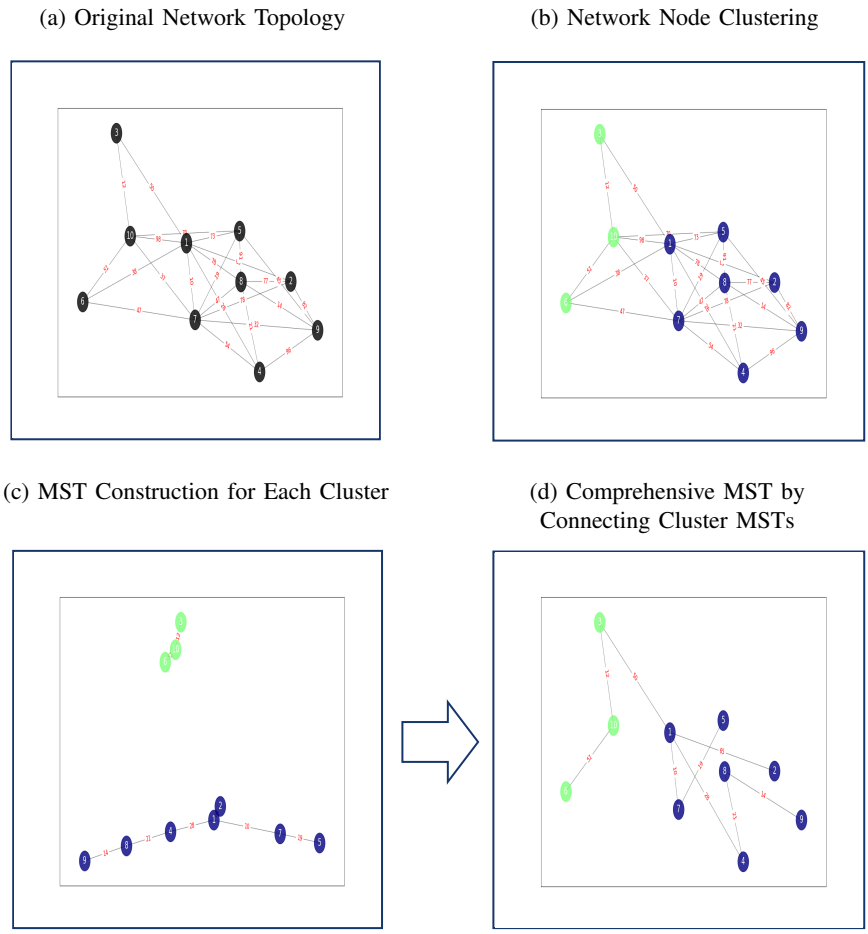

Fig. 2. An illustrative example showing the practical application of the proposed BNSF framework in a real-case scenario.

predict which node to target in advance. Moreover, it maintains the decentralization of the network since no complex hardware is required. This means that any node s_i can construct an MST for its cluster c_i without needing specialized equipment or high power.

The process of re-selecting a new leader is performed after a certain period to reduce network traffic. The BNSF framework allows new nodes to join the BC network only after the end of this period, so new nodes attempting to join the network are added to a waiting queue by the BNSF framework. New nodes in the waiting queue join the network when this period expires. Then, the network topology is once again divided into a set of clusters. Subsequently, a leader node is selected for each cluster to create a new MST for its cluster. If a node leaves the network, only the network topology of the cluster to which it belongs will be changed. Consequently, only a new leader for this cluster is re-selected. The new leader node then creates an MST for its cluster. Afterward, the BNSF framework connects it with the MSTs of other clusters. This makes the proposed framework dynamic in response to changes in network topology. The global MST of the entire network is then used in the NS process.

In **Step-3**, the leader announcement proposed by the BNSF framework can be described as follows:

Following the leader selection process, the BNSF notifies all nodes $s_i \in S_i$ in cluster c_i about the new leader by sending announcement messages to all of them. Additionally, it informs the new leader of their responsibility for creating the MST for their cluster and broadcasting it to all nodes within the cluster. This enables the nodes to choose the optimal neighbor for data exchange within the BC network through the provided MST.

D. Phase 3: MST Construction using Dijkstra's Algorithm

After announcing the cluster leader with their new responsibility for creating the MST using Dijkstra's Algorithm [49] and subsequently broadcasting the MST to all nodes in the cluster, the MST creation process can be described as follows:

In **Step-4**, each LN builds the MST network topology of its cluster c_i by collecting neighbor information $\mathcal{N}_{c_i}(s_i)$ for each node s_i in cluster c_i. When the nodes S_i receive the announcement message from the LN, every node transmits its neighbors' information $\mathcal{N}_{c_i}(s_i)$ back to the LN. The LN then uses the collected information to generate a comprehensive view of the network topology for its cluster c_i and constructs the MST for the cluster, as shown in Fig. 2(C). Subsequently, the BNSF framework connects the generated MSTs for each cluster with each other. Finally, a global MST network topology is created for the entire BC network, as illustrated in Fig. 2(D). This global MST can be utilized by network nodes S in the process of selecting neighbors for broadcasting data within

Algorithm 2 Construct MST for each Cluster using Dijkstra's Algorithm

Input: Network cluster $c_i = (S_i, E_i)$
Output: MST graph for cluster c_i.
// $d[_i]$ represents the distances between a node s_i and its parent node
// $p[_i]$ represents parent nodes for all nodes $s_i \in S_i$.
// Q represents a temporary list of node S_i

1: **procedure** COMPUTE_MST(c_i)
2: Initialize $d[s_1] \leftarrow 0$, $p[s_1] \leftarrow$ None and $Q \leftarrow S_i$
3: Initialize MST as an empty graph.
4: **for all** $s_i \in S_i$ except$\{s_1\}$ **do**
5: $d[s_i] \leftarrow \infty$
6: $p[s_i] \leftarrow$ None
7: **end for**
8: **while** Q is not empty **do**
9: $u \leftarrow$ node in Q with the minimum distance $d[u]$
10: Remove u from Q
11: **for all** neighbor s_i of u **do**
12: **if** weight$(u, s) < d[s_i]$ **then**
13: $d[s_i] \leftarrow$ weight(u, s_i)
14: $p[s_i] \leftarrow u$
15: **end if**
16: **end for**
17: **end while**
18: //Constructs the MST for cluster c_i
19: **for all** $s_i \in S_i$ **do**
20: **if** $p[s_i] \neq$ None **then**
21: Add edge $(s_i, p[s_i])$ with edge_weight $d[s_i]$ to the MST
22: **end if**
23: **end for**
24: **Return** MST.
25: **end procedure**

Algorithm 3 Construct Comprehensive MST (MST_{com})

Input: Clusters set **C**, Network graph G.
Output: MST_{com} : Comprehensive MST for all nodes.

1: $MST_{\text{com}} \leftarrow$ Empty Graph
2: $R = \{\}$. //R represents the set of root nodes for clusters **C**
3: **for** c_i in **C** **do**
4: $MST_{c_i} \leftarrow$ run COMPUTE_MST(c_i) in a separate thread x_i
5: Add root node of MST_{c_i} to R
6: Add nodes and edges of MST_{c_i} to MST_{com}
7: **end for**
8: Connect root nodes in R based on edges in G to form MST_{com}
9: **Return** MST_{com}

the network. Algorithm 2 provides a detailed view of how the leader node develops the network MST.

Algorithm 2 can be explained as follows:

First, select the first node s_1 from cluster c_i as the source node and initialize the set Q as the cluster's set of nodes (line 2). Then, initialize the distance set $d[s_i]$ and the parent set $p[s_i]$ for each node s_i in cluster c_i (lines 4 $\rightarrow$ 7). Subsequently, the algorithm starts with the source node s_1 and traverses multiple adjacent nodes to explore all interconnected edges. It identifies a collection of edges that form a tree encompassing every vertex, with each vertex representing a BC network node (lines 11 $\rightarrow$ 17). Finally, the distance and parent for each node are stored for use in constructing the MST topology (lines 19 $\rightarrow$ 23).

Afterward, the MST network topology of cluster c_i is constructed by acquiring the distances $d[s_i]$ to reach nodes from their parent nodes $p[s_i]$, for each node $s_i \in S_i$ within cluster c_i. A node without a parent node is considered the root node of the MST. Ultimately, the algorithm constructs the MST of cluster c_i using node predecessors $p[s_i]$ and their corresponding distances $d[s_i]$ (lines 20 $\rightarrow$ 24). Finally, the root node r of each MST cluster is connected. This results in a global MST_{com} for the entire BC network, as shown in Algorithm 3, which is then used in the process of selecting the optimal neighbor for data transmission in the network.

Algorithm 3 is used to compute the MST_{c_i} for each cluster c_i in parallel and build a comprehensive MST_{com} for the entire BC network topology by connecting all root nodes R of the clusters' MSTs.

The use of multiple threads $\mathcal{X}$ within Algorithm 3, also known as parallel computing [50], can accelerate the execution of the idea in several ways. By dividing a problem into smaller sub-problems that can be solved independently, multiple threads can work on different parts of the problem simultaneously, leading to faster execution times. Additionally, parallel computing can be used to take advantage of multiple CPUs or cores, resulting in further performance improvements. Therefore, parallel computing can be a powerful tool for accelerating the execution of ideas and achieving our goals more efficiently.

In **Step-5**, each LN broadcasts the MST_{com} to its cluster members. In cluster c_i, each node s_i derives its own optimal neighbors $MON(s_i)$ from the received MST_{com}. These optimal neighbors are then used by nodes in the NS process to transmit new blocks or transactions over the BC network.

E. Phase 4: Neighbor Selection (NS)

In **Step-6**, each node s_i in the BC network that receives the MST_{com} from the LN of its cluster, extracts its optimal neighbor nodes $MON(s_i)$ from the received MST_{com} by running Algorithm 4.

In **Step-7**, the proposed framework replaces the RNS approach with more informed selection criteria, resulting in improved metrics for the BC network, including the average time it takes to broadcast a new block or transaction and achieve lower finality times. Network nodes $s_i \in S$ use their $MON(s_i)$ in the NS process to optimally select neighbors, share data, and propagate new blocks and transactions. Each node s_i within a cluster c_i can determine the most suitable neighbors for transmitting and broadcasting information to both nodes within its cluster and nodes in other clusters. This selection process relies on the MST_{com} provided by the cluster leader, allowing each node to identify the optimal neighbors from the MON for data exchange. This proposed approach, built upon the utilization of MST_{com} rather than random selection, significantly improves network performance. It achieves

Algorithm 4 Find $MON\,(s_i)$ for each node s_i

Input: MST_{com}, Node s_i.
Output: MST Optimal Neighbors $MON\,(s_i)$ for node s_i.

```
procedure FIND_MON(MST_com, s_i)
    MON(s_i) = {}
    for s_j in MST_com do // search for node s_i in the MST_com
        if s_j = s_i then
            for all neighbor s_k of s_j do
                w ← weight(s_j, s_k)
                MON(s_i) = MON(s_i) ∪ {(s_k, w)}
            end for
            break
        end if
    end for
    return MON(s_i)
end procedure
```

this by decreasing the time required for broadcasting data or blocks within the network, enabling quicker data exchange among network nodes, reducing overall network bandwidth utilization, and effectively reducing the possibility of duplicate data. Consequently, the risk of transmitting the same information to a particular node multiple times is diminished since the selection of the same node from multiple neighbors is avoided during data exchange.

F. An Illustrative Example

This section provides an example that demonstrates how the MST works to construct a general MST_{com} for the entire permissionless public BC network. A random deployment of 10 nodes, denoted as $S = \{1, 2, 3, \ldots, 10\}$, is used and visualized in Fig. 2(A). The average number of neighbors is denoted by k, which equals 5.

Table III summarizes the edge weights between the 10 nodes. A value of 0 represents that there is no edge between these nodes.

TABLE III. EDGE WEIGHT BETWEEN THE 10 NODES

nodes	1	2	3	4	5	6	7	8	9	10
1	0	65	50	28	73	38	10	78	0	98
2	65	0	0	0	0	0	78	77	82	0
3	50	0	0	0	0	0	0	0	0	12
4	28	0	0	0	0	0	54	21	96	0
5	73	0	0	0	0	0	19	91	45	70
6	38	0	0	0	0	0	47	0	0	51
7	10	78	0	54	19	47	0	47	32	37
8	78	77	0	21	91	0	47	0	14	0
9	0	82	0	96	45	0	32	14	0	0
10	98	0	12	0	70	51	37	0	0	0

Algorithm 1 applies the agglomerative clustering algorithm to segment these nodes into two distinct clusters ($M = 2$): $c_1 = \{3, 6, 10\}$ and $c_2 = \{1, 2, 4, 5, 7, 8, 9\}$, as displayed in Fig. 2(b). Initially, each data node forms an individual cluster: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \ldots, \{10\}$. Algorithm 1 calculates distances between all cluster pairs using Euclidean distance, subsequently merging the closest clusters into single entities. For instance, if the closest clusters are $\{3\}$ and $\{10\}$, they merge into a new cluster: $\{3, 10\}$. This process iterates, adjusting the hierarchy to include $\{1\}$, $\{2\}$, …, $\{3, 10\}$, …, $\{9\}$. After each merge, distances between the new cluster and other clusters are recalculated. As an illustration, subsequent clusters such as $\{3, 10\}$ and $\{6\}$ merge into $\{3, 6, 10\}$. This iterative process persists until the desired number of clusters is achieved ($M = 2$). Now we have the two clusters $c_1 = \{3, 6, 10\}$ and $c_2 = \{1, 2, 4, 5, 7, 8, 9\}$.

Algorithm 2, "Construct MST for each Cluster," is used to calculate the MST for each of the two clusters individually. The resulting MSTs for the clusters are presented in Fig. 2(c). Consider the example of cluster $c_2 = \{1, 2, 4, 5, 7, 8, 9\}$. We begin by initiating the MST construction for this cluster. A queue Q is established, containing nodes 1, 2, 4, 5, 7, 8, and 9. The initial distances and parent node references for each node are outlined in Table IV at step 1. The MST construction starts with node 1 as the source node, assigned a distance of 0. The algorithm removes node 1 from Q and proceeds to evaluate neighboring nodes connected to 1 within the cluster c_2. Nodes $\{2, 4, 5, 7, 8\}$ exhibit edge weights to node 1 that are smaller than their initial distances (infinity). Consequently, the algorithm updates the distances and parents of these nodes as indicated in Table IV at step 2.

Subsequently, with $Q = \{2, 4, 5, 7, 8, 9\}$, node 7 emerges as the node with the minimum-weight edge to node 1, weighing 10. Among the remaining nodes in Q, $\{2, 4, 5, 8, 9\}$ possess edge weights to node 7. However, nodes 2 and 4 do not have their distances and parents updated due to their existing lower distances compared to the new edge weights. After removing node 7 from Q, the algorithm only updates the distances and weights of nodes $\{5, 8, 9\}$, as presented in Table IV at step 3. Continuing with $Q = \{2, 4, 5, 8, 9\}$, node 5 stands out as having the smallest edge weight to node 7, amounting to 19. First, Node 5 is then removed from Q. Nodes $\{8, 9\}$ exhibit edge weights to node 5, but due to higher weights of 91 and 45 for nodes 8 and 9 respectively, their distances and parents remain unchanged.

The progression leads to $Q = \{2, 4, 8, 9\}$. Among the remaining nodes, node 4 stands out for its smallest edge weight to node 1, measuring 28. Node 4 is removed from Q. Although nodes $\{8, 9\}$ also have edge weights to node 4, only node 8 has its parent and distance updated due to its lower weight of 21, as seen in Table IV at step 4. Continuing, with $Q = \{2, 8, 9\}$, node 8 displays the smallest edge weight to node 4, measuring 21. Node 8 is removed from Q. Among the remaining nodes in Q, node 2 has a higher weight than its current distance, leading to no update in its distance and parent. The algorithm proceeds to update only the parent and distance of node 9 in Table IV at step 5.

This leaves $Q = \{2, 9\}$. Node 9 holds the smallest edge weight to node 8, weighing 14. However, node 2 does not have its distance and parent updated due to its higher weight of 82. The algorithm only removes node 9 from Q. Finally, node 2 remains within Q, connected to node 1 with an edge weight of 65. The algorithm proceeds by removing node 2 from Q, resulting in an empty queue. As node 2 does not have any unvisited neighbors, the algorithm terminates.

The last step involves constructing the MST using the

stored distances and parent node values of the cluster nodes. Furthermore, the root node of the MST for each cluster c is denoted as r_c. Consequently, the root nodes for the clusters are $r_1 = \{3\}$ and $r_2 = \{1\}$. The union of all root nodes from the clusters is represented as $R = \{3, 1\}$.

TABLE IV. THE DISTANCES AND PARENT REFERENCES FOR CLUSTER c_2

	nodes	1	2	4	5	7	8	9
step 1	$p[s]$	None	None	None	None	None	None	None
	$d[s]$	0	∞	∞	∞	∞	∞	∞
step 2	$p[s]$	None	1	1	1	1	1	None
	$d[s]$	0	65	28	73	10	78	∞
step 3	$p[s]$	None	1	1	7	1	7	7
	$d[s]$	0	65	28	19	10	47	32
step 4	$p[s]$	None	1	1	7	1	4	7
	$d[s]$	0	65	28	19	10	21	32
step 5	$p[s]$	None	1	1	7	1	4	8
	$d[s]$	0	65	28	19	10	21	14

Ultimately, the BNSF framework establishes connections between all root nodes, resulting in a comprehensive MST_{com} for the entire BC Network, as illustrated in Fig. 2(d). This MST_{com} serves as the optimal pathway for data propagation within the BC network, ensuring efficient communication and dissemination of information among the nodes. Each node within the network extracts its optimal neighbors (MONs) from the comprehensive MST_{com} based on Algorithm 4.

In this example, the MONs of node 1 encompass a dictionary of nodes with their weight values $\{(2, 65), (3, 50), (4, 28), (7, 10)\}$, enabling seamless data exchange. Notably, these MONs correspond to nodes with the lowest weights compared to other neighbors in the original BC network, thereby speeding up the data transfer process throughout the network. Through the BNSF approach, the BC network achieves an efficient structure, facilitating secure and rapid data transmission across the entire network.

G. Complexity Analysis of Algorithms

Mainly BNSF consists of four algorithms, Algorithm 1 consists of two steps: filling the edge matrix A_E from BC network graph G and applying agglomerative clustering on A_E. The first step involves a loop and a nested loop, with $O(N^2)$ time complexity where N is the total BC network nodes. The second step has a loop with $O(N)$ time complexity. Inside this loop (line 12), Calculating pairwise distances between clusters $O(N^2)$. Thus, the overall complexity is roughly $O(N^3)$.

Algorithm 2 operates in two phases: the first phase calculates the shortest paths using Dijkstra's algorithm, which runs in $O(n_i^2)$ time. n denotes the number of nodes within cluster c_i, where n_i is a subset of N. The second phase constructs an MST using the calculated predecessor nodes. This phase requires considering all nodes and their corresponding predecessor edges, which results in an overall time complexity of $O(n_i)$. Therefore, the complexity of the entire algorithm is determined by Dijkstra's algorithm phase, which is typically $O(n_i^2)$.

Algorithm 3 concurrently constructs MSTs for multiple clusters. The complexity analysis centers on the function Compute_MST(c_i), which exhibits a time complexity of $\mathcal{O}(n_i^2)$, where n_i represents the count of nodes within cluster c_i, and i ranges from 1 to M. The overall complexity is bounded by $\max(\mathcal{O}(n_j^2))$, where j indicates the cluster index associated with the maximum number of nodes. This arises due to the parallel construction of MSTs across all clusters. This approach leverages the advantages of multi-threading while respecting the underlying cluster computation complexity.

The complexity analysis of Algorithm 4 is as follows: initializing $MON(s_i)$ as an empty set takes $O(1)$ time. The outer loop iterates through each node s_j in the MST_{com}, which depends on network nodes N. Inside, a loop iterates through each neighbor s_k of the current node s_j. The overall complexity is $O(N)$ (outer loop) * $O(k)$ (inner loop), where k represents the average number of neighbors for a node s_i. For sparse BC networks, complexity is nearly linear; for dense networks, it approaches $O(Nk)$.

IV. EXPERIMENTS AND RESULTS

This section includes the main experiments and evaluation of the proposed framework. The used network datasets, performance measures, and the conducted experiments are discussed in detail. Network data used in this study was generated by the simulator developed by [51]. The simulator built a random network topology using a random network model, namely the Barabási-Albert (BA) model [52]. It simulates nodes in real networks, which can be found in many natural and human-generated systems, including but not limited to the Internet, social networks, and the World Wide Web.

The simulation starts by generating a random BC network, where a miner node is selected at random as the source node for a data block. Subsequently, the source node shares the generated block with its neighboring nodes, and each neighbor continues this process with its own neighbors, creating a cascade effect. The simulation concludes once a block has successfully reached all nodes in the network.

The experiments were conducted on a DELL laptop featuring an Intel i5-5200U CPU (4 Cores, 2.2GHz), 12GB DDR3 RAM, a 250GB SSD Drive, and a Windows 10 operating system. The experimental results are checked and evaluated using the following performance metrics:

- Total Propagation Time (TP) (μs): is the time it takes for block data sent from a randomly selected miner node to propagate to all nodes within the network.
- MST_{com} calculation time (MST-CT) (sec): is the actual time required to build the MST_{com} network topology for the entire BC network.
- Number of exchanged blocks (NB): denotes the count of blocks exchanged between network nodes in order to broadcast the block sent from a randomly selected miner node, including redundant or repeated blocks that a node could receive from different neighbors until it reaches all network nodes.

The experiments conducted in this paper are classified into the following categories:

- Experiments 1 and 2 focus on analyzing the correlation between the BNSF parameters (Avg. no. of

neighbors k, no. of clusters M, and no. of nodes N) and performance metrics.

- Experiment 3 aims to enhance BNSF by employing various clustering algorithms such as Agglomerative, K-means, and Community Louvain.
- Experiment 4 involves comparing BNSF with other methods, specifically DONS, RTT-NS, and RNS.

Experiment 1

This experiment examines and discusses the effect of the average number of neighbors per node k on performance metrics TP and $MST\text{-}CT$, considering various numbers of nodes N (e.g., 500, 1000, and 1500). The number of clusters C is constant, set to 5. In Fig. 3(A), on the left-hand side, TP is plotted against k (e.g., 5, 10, 15, 20). TP decreases by up to 68.57% when k equals 20 and N equals 1500. In general, as k increases, TP decreases correspondingly. This is due to the increase in the number of potential neighbors for each node in the network, providing more options to select the best neighbor node and consequently build a better MST_{com} network with lower weight. The more neighbors a node has, the better the MST_{com} becomes. As a result, the process of broadcasting new blocks improves, as it relies on the best-created MST_{com}, leading to faster block propagation in the network.

In Fig. 3(B), on the right-hand side, $MST\text{-}CT$ is plotted against k. As observed, $MST\text{-}CT$ slightly reduces by 4.84% when k is set to 20, and N is 1500. As k increases, the change in $MST\text{-}CT$ remains minimal for every N of nodes, indicating that varying the number of neighbors for each node in the network does not significantly impact the calculation time required to construct the MST_{com} topology of the BC network. Conversely, the increase in the number of nodes N within the network significantly affects the MST_{com} calculation time $MST\text{-}CT$.

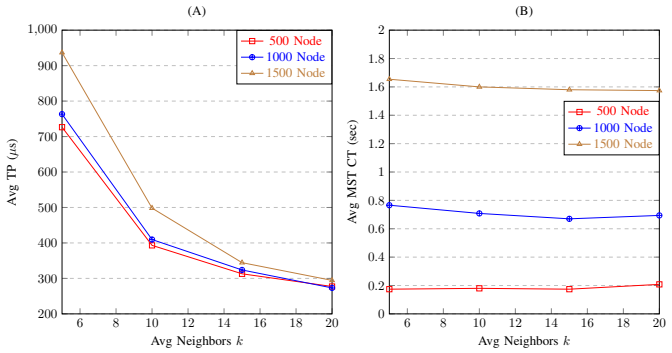

Fig. 3. Average number of neighbors (k) vs. (A) The average total propagation time (TP) and (B) the MST_{com} calculation time.

Experiment 2

In this experiment, the impact of the number of clusters M on performance metrics TP and $MST\text{-}CT$ is discussed while considering different numbers of nodes N (e.g., 500, 1000, and 1500). The average number of neighbors for every node k is constant, set to 15. In Fig. 4(A), TP is plotted against M (e.g., 2, 4, 6, 8, and 10). Generally, as M increases, the value of TP changes correspondingly but with irregular values. When N is equal to 500, it can be observed that with a significantly increased number of clusters M and a small number of nodes, there is a considerable increase in the propagation time TP. Consequently, it is better to choose a small number of clusters to match the small number of nodes. Furthermore, when N equals 1000 and 1500, a larger number of clusters can be selected due to the increased node count to obtain the best performance and the lowest propagation time TP.

In Fig. 4(B), MST_{com} calculation time ($MST\text{-}CT$) is plotted against the number of clusters M (e.g., 2, 4, 6, 8, and 10). When the number of clusters M increases, $MST\text{-}CT$ changes slightly for every N of nodes. Therefore, increasing or decreasing the number of network clusters does not significantly affect the calculation time required to construct the MST_{com} topology of the BC network. In contrast, $MST\text{-}CT$ is notably influenced by the increase in the number of nodes N within the network.

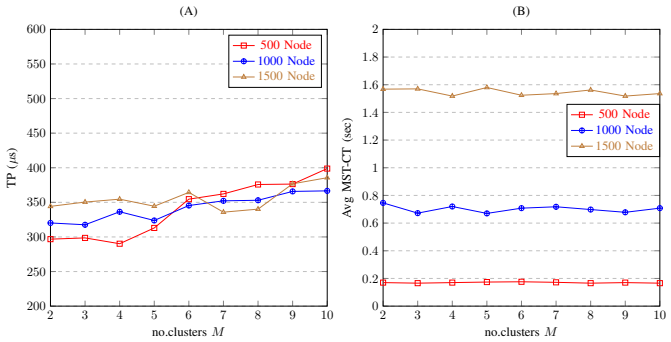

Fig. 4. Number of clusters (M) vs. (A) The average total propagation time (TP) and (B) MST_{com} calculation time.

Experiment 3

In this experiment, the BNSF framework was developed using different clustering algorithms to examine the efficiency of the proposed model, with a specific focus on agglomerative clustering. Two clustering algorithms, namely K-means and Community Louvain, were compared with the Agglomerative algorithm. K-means clustering [53] is a method that aims to group N nodes into M clusters by ensuring that each node is assigned to the cluster with the closest mean value, also known as the cluster center or centroid. On the other hand, Community Louvain is a clustering technique designed for large networks. It computes the best partition of the graph nodes by maximizing modularity using the Louvain heuristics. This results in the partition with the highest modularity achieved by the Louvain algorithm [54].

The number of clusters M is constant and set to 5 for both the agglomerative and K-means clustering methods. The configuration of the network size and the average number of neighbors per node is modified. This is done to demonstrate the advantages of the proposed framework with the Agglomerative clustering method in various real-life scenarios. Table V displays the best outcomes for the Total Propagation Time, highlighted in bold. As shown in Table V, the BNSF framework with Agglomerative clustering achieves the highest performance with the lowest propagation time. When

TABLE V. DEVELOPING THE BNSF FRAMEWORK WITH DIFFERENT CLUSTERING ALGORITHMS

Model	Network parameter	No.Nodes	Total Propagation Time (μs)					
	Avg.no.neighbors		**Agglomerative**		K-means		Community Louvain	
			Mean	**SD**	Mean	SD	Mean	SD
BA	10	500	**393.2**	**7.05**	1399.2	330.66	2007.8	226.71
		1000	**409.6**	**53.36**	1335.8	466.54	3030.4	367.69
		1500	**498.2**	**81.89**	1750.4	369.61	3486.2	245.82
	15	500	**313**	**60.99**	857.4	207.52	2112.6	405.23
		1000	**323.8**	**65.14**	1216	398.4	2567.4	273.12
		1500	**344.4**	**29.71**	1616.2	395.1	3185.6	352.66
	20	500	**277.2**	**21.73**	1107.6	184.96	1435.4	208.94
		1000	**273.2**	**31.42**	1190.4	238.18	1980.8	349.22
		1500	**294.4**	**33.32**	1091.2	241.54	2368.2	307.62

comparing it to other clustering algorithms like K-means, it outperforms by 73.02%, and when compared to Community Louvain, it outperforms by 87.57%, with k set to 20 and N set to 1500 in terms of TP.

Experiment 4

The proposed BNSF framework is assessed in terms of total propagation time and message complexity in comparison to commonly employed neighbor selection methods such as DONS, RNS, and NS based on local RTT. The four neighbor selection methods are compared under identical network conditions, with the same block originating from the same source node.

Several experiments have been conducted using a random network model, specifically the Barabási-Albert (BA) model. The number of nodes N and the average number of neighbors for every node k were varied to capture the behavior of the proposed framework under different network sizes.

The efficiency of BNSF was examined in terms of TP, $MST\text{-}CT$, and NB. The number of clusters M for BNSF equals 3 for $N = 500$, equals 5 for $N = 1000$, and equals 7 for $N = 1500$.

In this part of the experiment, the network size and the average number of neighbors for every node are varied with $k = 10$, 15, and 20 to illustrate the robustness of the proposed BNSF framework in diverse real-life scenarios. The results obtained from all algorithms, along with the outcomes of different simulation scenarios, are presented in Table VI.

According to Table VI, the BNSF framework and the DONS algorithm do not have redundant blocks when exchanging information between nodes in the BC network, as nodes keep track of the replicated blocks they receive. The more redundant blocks, the more blocks are exchanged between nodes in the network, resulting in higher overhead on communication links and computational burden at the node level. Consequently, this leads to an elevated total propagation time. However, the BNSF framework outperforms the other algorithms, namely RNS and RTT-NS, in terms of the number of blocks exchanged within the network.

The proposed BNSF framework also outperforms the DONS algorithm on other points like the propagation time of blocks within the network (TP) and the duration needed to construct the MST_{com} of the BC network ($MST\text{-}CT$). Furthermore, according to Table VI, the proposed BNSF framework outperforms the other algorithms like DONS (by 51.14%), RTT-NS (by 99.16%), and RNS (by 99.95%) in terms of TP, when k equals 20, and N equals 1500.

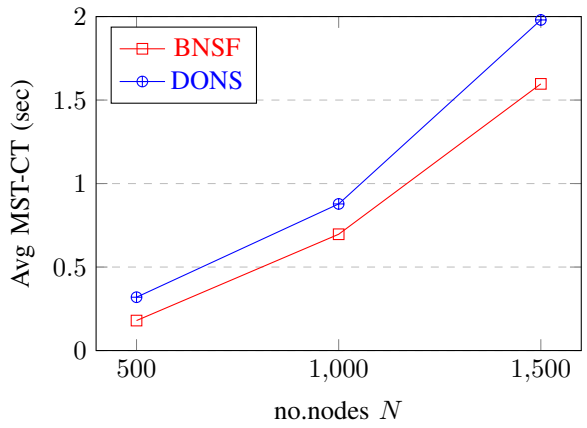

Fig. 5. Average $MST\text{-}CT$ for BNSF and DONS with Different Numbers of Nodes N.

In Fig. 5, the proposed BNSF framework is compared with the DONS algorithm in terms of $MST\text{-}CT$, which is plotted against the number of nodes N (e.g., 500, 1000, and 1500). As observed, the average $MST\text{-}CT$ achieved by the proposed BNSF framework is 28.48% lower than that of the DONS algorithm. These results demonstrate the superior performance of BNSF over the DONS algorithm.

When increasing the number of clusters, the $MST\text{-}CT$ should exhibit variations for different node counts N (e.g., 500, 1000, and 1500), depending on the network topology and the distribution of nodes within clusters. Thus, the calculation time required for constructing the MST_{com} topology of the BC network is significantly influenced by the number of clusters in the network and its size, reducing it to approximately 27.92% below that of the DONS algorithm. Computing the MST for each cluster of nodes in separate threads will result in minimizing the calculation time for the complete BC network's MST_{com}.

V. CONCLUSIONS AND FUTURE WORK

The paper introduces an improved dynamic neighbor selection BNSF framework to tackle neighbor selection and scal-

TABLE VI. PERFORMANCE OF THE PROPOSED BNSF FRAMEWORK AGAINST DONS, RTT-NS, AND RNS METHODS ON A RANDOMLY GENERATED NETWORK MODEL (BA) WITH VARYING SIZES

Model	Network parameter	No.Nodes	Total Propagation Time (μs)							
	Avg.no.neighbors		**BNSF**		DONS		RTT-NS		RNS	
			Mean	**SD**	Mean	SD	Mean	SD	Mean	SD
BA	10	500	**330.2**	**30.10**	590.33	94.2	18837	723.82	167146.67	18073.32
		1000	**409.6**	**53.36**	894	236.99	38864.33	1554.55	335111.33	30211.1
		1500	**536**	**58.51**	999.67	157.05	60596	3977.12	546851.67	63705.87
	15	500	**298.6**	**34.46**	499	113.92	14123	1182.76	135582.33	20082.65
		1000	**323.8**	**65.14**	523.67	84.39	29934.33	2358.39	389176	19521.17
		1500	**335.8**	**82.33**	892	94.16	45594.33	1127.41	586644.33	88479.45
	20	500	**263.4**	**17.83**	390.4	59.81	10923.8	1193.06	135820	22940.17
		1000	**273.2**	**31.42**	498.4	60.43	23308.2	722.98	327798	54554.98
		1500	**298.8**	**42.76**	611.6	57.98	35393.6	2042.82	548139.4	64174.62
			Avg Number of exchanged blocks (NB)							
			BNSF		DONS		RTT-NS		RNS	
			Mean	**SD**	Mean	SD	Mean	SD	Mean	SD
BA	10	500	**500**	**0**	500	0	674	13.64	6088.67	702.89
		1000	**1000**	**0**	1000	0	1340.33	52.64	12047.67	1138.42
		1500	**1500**	**0**	1500	0	2060	68.59	20227.67	2491.43
	15	500	**500**	**0**	500	0	610.67	23.61	4929.33	734.42
		1000	**1000**	**0**	1000	0	1266.33	49.88	14730	700.33
		1500	**1500**	**0**	1500	0	1884.67	16.78	21983	3551.55
	20	500	**500**	**0**	500	0	578.8	19.03	5071	936.52
		1000	**1000**	**0**	1000	0	1166	8.44	12402.2	2208.76
		1500	**1500**	**0**	1500	0	1768.2	26.76	20802.8	2561.74

ability issues in public blockchain networks. This framework reduces block propagation time, enhancing block or transaction throughput compared to traditional methods. As blockchain networks expand, the BNSF framework adapts by dividing the network topology into clusters and utilizing a multi-leader node approach. Multi-threading is employed to compute the MST of clusters concurrently, thereby enhancing scalability and ensuring efficient neighbor selection for faster and more streamlined block propagation.

The proposed BNSF framework demonstrates a significant reduction in total block propagation time, with a decrease of up to 68.57% when the average number of neighbors is 20 for each node and the total number of network nodes is 1500. Utilizing agglomerative clustering achieves superior performance, outperforming K-means by 73.02% and Community Louvain by 87.57% in total block propagation time, with similar network parameters.

The results of the proposed work showed a significant improvement in block propagation for networks of various sizes, surpassing state-of-the-art methods. The proposed BNSF framework is also effective in large-scale networks with a high node count. These experiments also revealed the BNSF framework's exceptional performance compared to alternative neighbor selection methods such as DONS, RNS, and RTT-NS. Furthermore, it decreases the overall time for block propagation, surpassing DONS by 51.14%, RTT-NS by 99.16%, and RNS by 99.95%. Additionally, the BNSF framework achieves an average MST_{com} calculation time of 27.92% lower than the DONS algorithm. Finally, it ensures the absence of redundant blocks during information exchange among nodes in the BC network.

In future work, further investigation will be conducted into alternative clustering methods for network partitioning and the exploration of alternative protocols for identifying leader nodes within clusters to enhance the efficiency of the BNSF framework. The impact of these choices on the framework's performance and efficiency will be thoroughly examined. Additionally, potential upgrades to the BNSF framework to serve as a comprehensive gossip and consensus protocol for public blockchain networks will be explored.

REFERENCES

[1] O. Akanfe, D. Lawong, and H. R. Rao, "Blockchain technology and privacy regulation: Reviewing frictions and synthesizing opportunities," *International Journal of Information Management*, vol. 76, p. 102753, 2024.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

[3] G. Zhang, F. Pan, Y. Mao, S. Tijanic, M. Dang'ana, S. Motepalli, S. Zhang, and H.-A. Jacobsen, "Reaching consensus in the byzantine empire: A comprehensive review of bft consensus algorithms," *ACM Computing Surveys*, vol. 56, no. 5, pp. 1–41, 2024.

[4] L. K. Ramasamy and F. Khan, "Utilizing blockchain for a decentralized database of educational credentials," in *Blockchain for Global Education*, pp. 19–35, Springer, 2024.

[5] A. K. Tyagi, "Decentralized everything: Practical use of blockchain technology in future applications," in *Distributed Computing to Blockchain*, pp. 19–38, Elsevier, 2023.

[6] B. Wen, Y. Wang, Y. Ding, H. Zheng, B. Qin, and C. Yang, "Security and privacy protection technologies in securing blockchain applications," *Information Sciences*, vol. 645, p. 119322, 2023.

[7] J. Liu and J. Wu, "A comprehensive survey on blockchain technology and its applications," *Highlights in Science, Engineering and Technology*, vol. 85, pp. 128–138, 2024.

[8] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, "Blockchain for 5g-enabled iot for industrial automation: A systematic review, solutions, and challenges," *Mechanical systems and signal processing*, vol. 135, p. 106382, 2020.

[9] D. Das, S. Banerjee, K. Dasgupta, P. Chatterjee, U. Ghosh, and U. Biswas, "Blockchain enabled sdn framework for security management in 5g applications," in *Proceedings of the 24th International Conference on Distributed Computing and Networking*, pp. 414–419, 2023.

[10] S. Onopa and Z. Kotulski, "State-of-the-art and new challenges in 5g networks with blockchain technology," *Electronics*, vol. 13, no. 5, p. 974, 2024.

[11] L. Tan, H. Xiao, K. Yu, M. Aloqaily, and Y. Jararweh, "A blockchain-empowered crowdsourcing system for 5g-enabled smart cities," *Computer Standards & Interfaces*, vol. 76, p. 103517, 2021.

[12] Z. Ullah, M. Naeem, A. Coronato, P. Ribino, and G. De Pietro, "Blockchain applications in sustainable smart cities," *Sustainable Cities and Society*, p. 104697, 2023.

[13] S. F. A. Shah, T. Mazhar, T. Al Shloul, T. Shahzad, Y.-C. Hu, F. Mallek, and H. Hamam, "Applications, challenges, and solutions of unmanned aerial vehicles in smart city using blockchain," *PeerJ Computer Science*, vol. 10, p. e1776, 2024.

[14] P. Danzi, A. E. Kalør, Č. Stefanović, and P. Popovski, "Delay and communication tradeoffs for blockchain systems with lightweight iot clients," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2354–2365, 2019.

[15] W. A. Al-Nbhany, A. T. Zahary, and A. A. Al-Shargabi, "Blockchain-iot healthcare applications and trends: A review," *IEEE Access*, 2024.

[16] L. N. CheSuh, R. Á. F. Díaz, J. M. A. Perez, C. B. Cuellar, and H. A. Moretón, "Improve quality of service for the internet of things using blockchain & machine learning algorithms.," *Internet of Things*, p. 101123, 2024.

[17] L. Jiang and X. Zhang, "Bcosn: A blockchain-based decentralized online social network," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1454–1466, 2019.

[18] F. Mlika, W. Karoui, and L. B. Romdhane, "Blockchain solutions for trustworthy decentralization in social networks," *Computer Networks*, p. 110336, 2024.

[19] A. Gunawan, R. Richard, S. C. Chang, and S. Shilvi, "A review of data security of blockchain applications in social media," in *AIP Conference Proceedings*, vol. 3026, AIP Publishing, 2024.

[20] A. K. Tyagi and S. Tiwari, "The future of artificial intelligence in blockchain applications," in *Machine Learning Algorithms Using Scikit and TensorFlow Environments*, pp. 346–373, IGI Global, 2024.

[21] A. M. S. Saleh, "Blockchain for secure and decentralized artificial intelligence in cybersecurity: A comprehensive review," *Blockchain: Research and Applications*, p. 100193, 2024.

[22] A. Kuznetsov, P. Sernani, L. Romeo, E. Frontoni, and A. Mancini, "On the integration of artificial intelligence and blockchain technology: A perspective about security," *IEEE Access*, 2024.

[23] A. I. Sanka and R. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Applications*, vol. 195, p. 103232, 2021.

[24] I. S. Rao, M. Kiah, M. M. Hameed, and Z. A. Memon, "Scalability of blockchain: a comprehensive review and future research direction," *Cluster Computing*, pp. 1–24, 2024.

[25] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th usenix security symposium (usenix security 16)*, pp. 279–296, 2016.

[26] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari, and Y. Cao, "A survey on blockchain technology: Evolution, architecture and security," *Ieee Access*, vol. 9, pp. 61048–61073, 2021.

[27] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–35, 2023.

[28] N. Shi, L. Tan, W. Li, X. Qi, and K. Yu, "A blockchain-empowered aaa scheme in the large-scale hetnet," *Digital Communications and Networks*, vol. 7, no. 3, pp. 308–316, 2021.

[29] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *Journal of Network and Computer Applications*, vol. 209, p. 103539, 2023.

[30] R. Antwi, J. D. Gadze, E. T. Tchao, A. Sikora, H. Nunoo-Mensah, A. S. Agbemenu, K. O.-B. Obour Agyekum, J. O. Agyemang, D. Welte, and E. Keelson, "A survey on network optimization techniques for blockchain systems," *Algorithms*, vol. 15, no. 6, p. 193, 2022.

[31] L. Zhang, T. Wang, and S. C. Liew, "Speeding up block propagation in bitcoin network: Uncoded and coded designs," *Computer Networks*, vol. 206, p. 108791, 2022.

[32] C. Li, J. Zhang, X. Yang, and L. Youlong, "Lightweight blockchain consensus mechanism and storage optimization for resource-constrained iot devices," *Information Processing & Management*, vol. 58, no. 4, p. 102602, 2021.

[33] G. Saldamli, C. Upadhyay, D. Jadhav, R. Shrishrimal, B. Patil, and L. Tawalbeh, "Improved gossip protocol for blockchain applications," *Cluster Computing*, vol. 25, no. 3, pp. 1915–1926, 2022.

[34] N. El Rharbi, H. Atteriuas, A. Younes, A. Harchaoui, and O. Izem, "A comparative study of the recent blockchain consensus algorithms," in *E-Learning and Smart Engineering Systems (ELSES 2023)*, pp. 316–327, Atlantis Press, 2024.

[35] N. Loizou and P. Richtárik, "Revisiting randomized gossip algorithms: General framework, convergence rates and novel block and accelerated protocols," *IEEE Transactions on Information Theory*, vol. 67, no. 12, pp. 8300–8324, 2021.

[36] G. Danner, I. Hegedűs, and M. Jelasity, "Improving gossip learning via limited model merging," in *International Conference on Computational Collective Intelligence*, pp. 351–363, Springer, 2023.

[37] W. Bi, H. Yang, and M. Zheng, "An accelerated method for message propagation in blockchain networks," *DOI: 10.48550/arXiv.1809.00455*, 2018.

[38] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *Ieee Access*, vol. 8, pp. 16440–16455, 2020.

[39] K. Wang and H. S. Kim, "Fastchain: Scaling blockchain system with informed neighbor selection," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 376–383, 2019.

[40] H. Baniata, A. Anaqreh, and A. Kertesz, "Dons: Dynamic optimized neighbor selection for smart blockchain networks," *Future Generation Computer Systems*, vol. 130, pp. 75–90, 2022.

[41] W. Hao, J. Zeng, X. Dai, J. Xiao, Q. Hua, H. Chen, K.-C. Li, and H. Jin, "Blockp2p: Enabling fast blockchain broadcast with scalable peer-to-peer network topology," in *International Conference on Green, Pervasive, and Cloud Computing*, pp. 223–237, Springer, 2019.

[42] Y. Aoki and K. Shudo, "Proximity neighbor selection in blockchain networks," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 52–58, 2019.

[43] C. Santiago and C. Lee, "Accelerating message propagation in blockchain networks," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 157–160, IEEE, 2020.

[44] V. Deshpande, H. Badis, and L. George, "Efficient topology control of blockchain peer to peer network based on sdn paradigm," *Peer-to-Peer Networking and Applications*, vol. 15, no. 1, pp. 267–289, 2022.

[45] H. Vu and H. Tewari, "An efficient peer-to-peer bitcoin protocol with probabilistic flooding," in *Emerging Technologies in Computing: Second International Conference, iCETiC 2019, London, UK, August 19–20, 2019, Proceedings*, pp. 29–45, Springer, 2019.

[46] G. Fortino, F. Messina, D. Rosaci, and G. M. Sarnè, "Using trust measures to optimize neighbor selection for smart blockchain networks in the iot," *IEEE Internet of Things Journal*, 2023.

[47] E. K. Tokuda, C. H. Comin, and L. d. F. Costa, "Revisiting agglomerative clustering," *Physica A: Statistical Mechanics and its Applications*, vol. 585, p. 126433, 2022.

[48] P. Dawyndt, H. D. Meyer, and B. D. Baets, "The complete linkage clustering algorithm revisited," *Soft Computing*, vol. 9, pp. 385–392, 2005.

[49] J.-C. Chen, "Dijkstra's shortest path algorithm," *Journal of formalized mathematics*, vol. 15, no. 9, pp. 237–247, 2003.

[50] R. Saavedra-Barrera, D. Culler, and T. Von Eicken, "Analysis of multithreaded architectures for parallel computing," in *Proceedings of the second annual ACM symposium on Parallel algorithms and architectures*, pp. 169–178, 1990.

[51] H. Baniata, A. Anaqreh, and A. Kertesz, "Dons simulator." https://github.com/HamzaBaniata/DONS_simulator/, 2022.

[52] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[53] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[54] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.