A Distributed Algorithm to Critical Node Identification in IoT Networks

Ali Lalouci, and Zoubeyr Farah

***Abstract*—The increasing integration of Internet of Things (IoT) networks in various sectors has intensified the need for advanced security mechanisms to mitigate vulnerabilities. Among these, critical node detection has emerged as a key strategy to improve network resilience. In this paper, we introduce a distributed algorithm to address the Component-Cardinality-Constrained Critical Node Problem (3C-CNP) in IoT networks, a variant of the widely studied Critical Node Detection Problem (CNDP). The 3C-CNP involves identifying the minimal subset of nodes whose removal results in the fragmentation of the network into connected components, each containing no more than a specified number of nodes. To the best of our knowledge, this is the first distributed solution proposed for this variant. We provide a detailed description of the algorithm and analyze its computational complexity. Furthermore, we validate its performance through extensive simulations using CupCarbon, a widely recognized tool for designing and simulating IoT networks.**

***Index Terms*—Critical nodes, Network connectivity, IoT networks, Distributed computation, CupCarbon IoT simulator.**

I. Introduction

An IoT network is a collection of nodes, each representing a distinct device or entity equipped with sensors, actuators, and computing capabilities. It is used for different purposes in healthcare, smart homes, smart cities, industrial automation, and entertainment. The communication link between nodes is provided by various types of communication network, including wireless sensor networks, ZigBee, Wi-Fi, mobile ad hoc networks, etc. However, one of the main requirements in all kinds of networks is reliable communication. In a wireless network, losing some special nodes can significantly degrade the network's performance, such as network connectivity. Generally, these nodes, known as critical nodes, have numerous adverse effects on the network, and detecting them is a challenging problem.

In graphs or networks, the CNDP seeks to determine a subset of nodes whose removal allows maximally destroy the network connectivity regarding certain predefined metric. The formal definition of the CNDP is usually given as follows: the problem takes as input a graph $G = (V, E)$ and a predefined connectivity metric denoted by σ, and returns as output a set of nodes $S \subseteq V$ whose removal optimizes an objective function denoted by $f(\sigma)$. This problem finds its applications in several fields, including social network analysis [1], [2], network immunization [3], transportation engineering [4], [5], telecommunications [6], military strategic planning [7], IoT networks analysis [8], [57], network security [9], [10], and many others. In terms of network analysis, the determination of the critical node has a direct meaning in network security. Therefore, CNDP has attracted a lot of attention in recent years, and many connectivity metrics and variants of this problem have been defined and studied in the literature. These variants depend on how the network is disconnected once the nodes have been removed, including: the Critical Node Problem (CNP) [11], [12], the problem of Maximizing the Number of connected components(MaxNum) [13], [14], the problem of Minimizing the Maximal Component size(MinMaxC) [15], [16], the $\beta-$vertex disruptor problem [17], [18], the Component-Cardinality-Constrained Critical Node Problem (3C-CNP) [19], [20], etc. However, it is well-known that most variants of the CNDP are NP-hard, even when restricted to particular classes of graphs.

Historically, CNDP originates from the work of Borgatti et al. [1], where the authors proposed several methods and metrics to identify the most important nodes in a network, known as key players. Building on this foundation, Arulselvan et al. [2] focused on the pairwise connectivity metric and formally defined the critical node problem as the task of identifying a set of nodes whose removal minimizes pairwise connectivity (or path survivability) in the network. They demonstrated the NP-completeness of the recognition version of CNDP on general graphs, developed a linear programming model to solve the problem effectively, and proposed a greedy algorithm that outperformed other methods on randomly generated instances. Subsequently, Di-Summa et al. [12] presented complexity results and developed algorithms for various versions of the CNDP on trees, incorporating edge costs and node weights. They demonstrated that the CNDP on trees remains NP-complete when general connection costs are specified. However, for instances with unit connection costs, they proposed a polynomial-time algorithm based on a dynamic programming approach. Likewise, Addis et al. [21] demonstrated the NP-completeness of the CNDP for several classes of graphs, including split graphs, bipartite graphs, and complement bipartite graphs. In contrast, they presented a polynomial-time algorithm for identifying critical nodes in graphs with a

Manuscript received January 20, 2025; revised February 15, 2025. Date of publication March 10, 2025. Date of current version March 10, 2025. The associate editor prof. Teodoro Montanaro has been coordinating the review of this manuscript and approved it for publication.

A. Lalouci is with the Université de Bejaia, Faculté des Sciences Exactes, Département d'Informatique, 06000, Bejaia, Algeria (e-mail: ali.lalouci@univ-bejaia.dz, ali.lalouci@centre-univ-mila.dz). Z. Farah is with the Université de Bejaia, Faculté des Sciences Exactes, Laboratoire LIMED, 06000, Bejaia, Algeria (e-mail: zoubeyr.farah@univbejaia.dz).

Digital Object Identifier (DOI): 10.24138/jcomss-2024-0125

bounded treewidth. The NP-completeness of the critical node problem was also established for Unit-Disk graphs and Power-Law graphs by Shen et al. [22]. Additionally, the authors proposed effective greedy algorithms to identify both critical nodes and links, with the results being applied to assess network vulnerability. Guettiche and Kheddouci [5] focused on the critical nodes and links problem, using the shortest path metric. They proposed algorithms designed to assess the reliability of transportation networks, further extending the applicability of CNDP methodologies in infrastructure analysis. Shen et al. [14] investigated new variants of the CNDP based on two distinct metrics: maximizing the number of connected components and minimizing the size of the largest component. They proposed polynomial-time algorithms for solving these variants on series-parallel graphs, $k-$hole graphs, and trees. Similarly, Lalou et al. [20] studied the critical node problem with respect to bounding the size of the largest connected component and introduced a new variant, termed the Component-Cardinality-Constrained Critical Node Problem (3C-CNP). For a comprehensive overview of CNDP variants and their applications, readers are encouraged to refer to the survey by Lalou et al. [23]. Recently, several research works have been published, like Hosteins et al. [24], which needed that critical nodes set to be connected, whereas Di-Summa and Faruk [25] has initiate the hybrid problem of Critical node/edge detection problems, which seeks to determine a set of nodes and/or edges of given cardinality, whose removal maximally destroyed the network connectivity throw connected pairwise minimizing. More recently, another survey on critical node detection with their applications and challenges is performed in 2023 [26].

In the realm of IoT networks, identifying critical nodes is a fundamental step for enhancing the reliability and resilience of communication systems. This recognition serves as the foundation for various studies addressing the CNDP. For instance, in [27], the authors examined the CNDP to strengthen network defense in MANET-IoT networks. They proposed a method termed Dynamic Critical Node Identification (DCNI) to identify critical nodes. The complexity of the proposed DCNI is approximately $O(m * n^2)$, where n is the number of nodes and m represents the number of links in the network. In [28], the authors investigate the CNDP in Industrial Wireless Sensor and IoT networks. They proposed a two-phase algorithm: Phase I employs a distributed approach for critical node detection (Algorithm 1), while Phase II enhances node resilience through a centralized approach (Algorithm 2). The proposed algorithms require $O(log(n))$ time for convergence and $O(\delta(logn))$ for Critical Node detection, n represents the number of IoT devices, and δ is the cost required to forward the message. Recently, Ishfaq et al. [29] have addressed the CNDP in an IoT network. They developed an efficient and minimalistic integer linear programming solution, designed to optimize the cost of an attack on the network. Their experimental results demonstrate that this approach is both fast and scalable, suitable for large-scale infrastructure systems. The solution achieves one of two objectives: maximizing the damage caused to the network within a fixed budget or minimizing the cost of an attack that causes a desired amount of damage. Ugurlu et al. [8] present a survey on critical node detection methods in IoT along with their applications and challenges.

Consider the 3C-CNP variant of CNDP. Numerous centralized algorithms have been proposed in the literature to address this variant in both general graphs and specific graph classes (see, for example, [31], [19], [17], [32] and the references therein). These approaches encompass integer programming formulations, branch-and-cut algorithms, approximation methods, and evolutionary algorithms. Although these approaches offer reasonable computational complexity, their applicability to unreliable platforms, such as wireless sensor networks (WSNs) and IoT networks, remains limited. Moreover, they are not well-suited for modern distributed systems and simulation frameworks such as NS-3, Cooja, Tossim, Riverbed, and CupCarbon. In particular, CupCarbon is recognized as a highly efficient tool for analyzing IoT networks from a graph-theoretic perspective [58].

On the other hand, tree structures offer compelling advantages for IoT networks, including enhanced energy efficiency, scalability, ease of management, fault tolerance, and improved data handling capabilities. These attributes make tree topologies a preferred choice for many IoT applications, ensuring both reliable and efficient network performance.

In this paper, we address the 3C-CNP, a variant of the CNDP within such networks. We propose a Distributed Algorithm for the Component-Cardinality-Constrained Critical Node Problem (DA3C-CNP) specifically designed to identify critical nodes in tree-structured IoT networks. In our proposed approach, execution begins with the leaf nodes, which broadcast messages to their respective parent nodes. Each parent node aggregates the received messages and compares the sum against a predefined upper bound. If the sum exceeds this threshold, the parent node is marked as a critical node; otherwise, it forwards the accumulated sum to its parent in the hierarchy. This process continues iteratively, propagating the information upward until it reaches the root of the tree. Our proposed algorithm is designed to function in both anonymous (identifier-free) and non-anonymous IoT networks. It guarantees convergence within $\Delta - 1$ iterations per node and requires the exchange of only n messages, ensuring high communication efficiency.

The main contributions of this work are as follows:

1) We propose the first distributed algorithm designed specifically for solving the 3C-CNP, a variant of the widely recognized CNDP.
2) We conduct a detailed complexity analysis of the proposed algorithm and compare its performance with other centralized approaches addressing the same variant of the problem.
3) To demonstrate the practical applicability of the proposed algorithm, we adapt it to an IoT network context and evaluate its effectiveness in a representative scenario using the CupCarbon simulator.

The remainder of the paper is structured as follows. Section II provides the definition of the 3C-CNP variant and discusses its related work. In Section III, we present the proposed

distributed algorithm, DA3C-CNP, including a detailed description of the variables and primitive functions used, followed by an explanation of the algorithm's execution process. Section IV presents the experimental setup and simulation results, highlighting the performance of the algorithm using the CupCarbon IoT simulator. Finally, Section V concludes the paper and outlines potential directions for future research.

II. Problem Statement and Related Works

This section provides an overview of the 3C-CNP and its relevance to IoT networks. We also review the existing literature on this variant of the CNDP, highlighting various approaches, algorithms, and their applicability to different network structures.

A. Problem definition

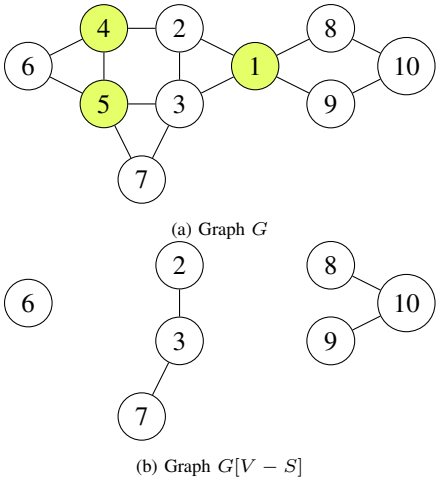
Let $G = (V, E)$ be an unweighted connected graph representing a network, where V is the set of n nodes and $E \subseteq V * V$ is the set of m edges. Given a subset of nodes $S \subseteq V$, let $G[S]$ and $G[V - S]$ denote the subgraphs of G induced by S and $V - S$, respectively.

The 3C-CNP seeks to find the minimal subset of nodes within a graph G, whose deletion results in a set of connected components, each possessing a cardinality not exceeding a specified integer bound B. A formal definition of 3C-CNP is as follows:

Component-Cardinality-Constrained Critical Node Problem (3C-CNP)

Input: A graph $G(V, E)$ and an integer B.
Output: A minimum set of nodes $S \subseteq V$, such that $|h| \leq B$ for each subset of connected nodes $h \in G[V - S]$.

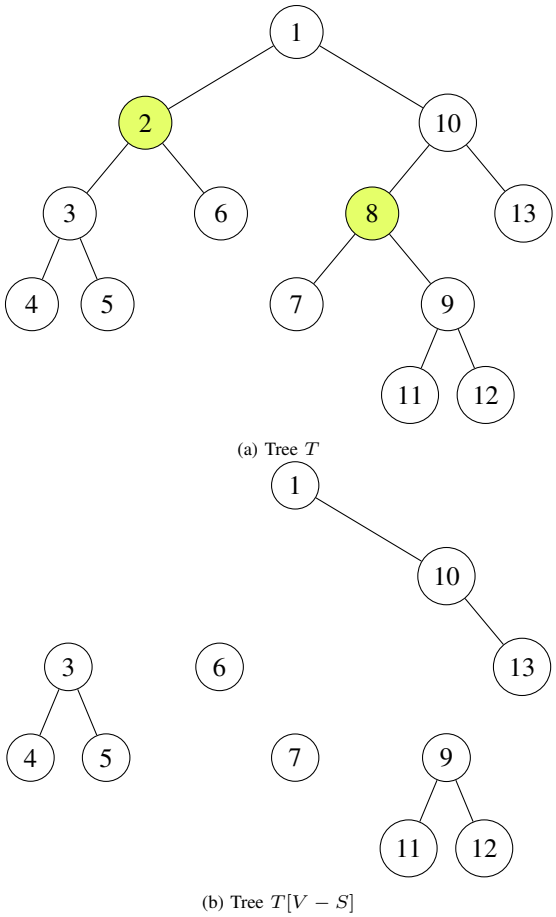
In Figure. 1, we give an example of a graph $G = (V, E)$ with ten nodes and an integer bound $B = 3$. The set of critical nodes $S = \{1, 4, 5\}$, and the number of connected components is 3. In Figure. 1a, the shaded nodes represent a set of critical nodes. Figure. 2 displays an example of a tree $T = (V, E)$ with 13 nodes. In Figure. 2a, the shaded nodes represent a set of critical nodes.



(a) Graph G

(b) Graph $G[V - S]$

Fig. 1. 3C-CNP applied to a general graph



(a) Tree T

(b) Tree $T[V - S]$

Fig. 2. 3C-CNP applied to a Tree

B. Applications

The 3C-CNP has become a significant research focus due to its wide range of applications across various disciplines, including social network analysis, biology, communication networks, and network reliability. In this subsection, we present the most important applications considered in the literature, as illustrated in Figure 3.

1) *Social network analysis:* Analyzing a social network often involves detecting communities within the network structure [36], [38]. For instance, if a threshold is specified for community size, one can identify network communities accordingly. This is particularly relevant for terrorist networks, where the sizes of terrorist groups (or communities) may be known. In such scenarios, critical nodes are those that connect different communities.

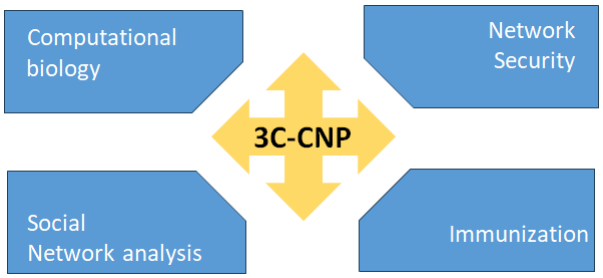


Fig. 3. Examples of 3C-CNP applications

Thus, solving the 3C-CNP can be used to identify these terrorist groups.

2) *Computational biology:* Biological organisms, which consist of interconnected proteins that interact to form a protein interaction network, can be represented by a graph in which the nodes are proteins and the edges represent the interactions between them. The application of 3C-CNP can be used to identify the minimal number of proteins whose destruction would neutralize the harmful organism[39].
3) *Immunization:* In the event of undesirable occurrences, such as viruses spreading in networks, epidemics, or the propagation of malware, 3C-CNP helps identify individuals (or devices) whose immunization can stop the spread of the epidemic. Additionally, it helps locate the source of the diffusion [40], [41]
4) *Network reliability:* Ensuring the security of network applications against malicious behavior is a fundamental design consideration. The application of the 3C-CNP can contribute to improved network security by providing a metric for network vulnerability. Specifically, the 3C-CNP assesses vulnerability based on the principle that a network requiring the removal of more critical nodes for partitioning is considered less vulnerable. In contrast, the fewer nodes that need to be removed, the more easily the network can be compromised[20], [35].

C. Related Works

In the introduction, we presented a comprehensive overview of the research related to CNDP in general. In this subsection, we provide a detailed examination of the studies and methodologies that specifically address the 3C-CNP variant.

3C-CNP is a variant of the Cardinality-Constrained Critical Node Problem (CC-CNP), introduced by Arulselvan et al. in 2011 [30], for identifying critical nodes in telecommunication networks. Their work focuses on minimizing the number of vertices whose removal results in disconnected components, constrained by a predefined cardinality. They also proved that the CC-CNP is NP-complete on general graphs. Subsequently, Lalou et al. [20] expanded on this concept by introducing and analyzing a new variant of the problem, referred to as 3C-CNP. In this variant, the objective is to identify a minimal set of nodes whose removal ensures that the order of each connected component in the resulting graph remains within a specified bound. they demonstrated the NP-completeness of the recognition version of the problem for general graphs, and even when restricted to bounded-degree graphs of maximum degree $\Delta = 4$.

Several centralized approaches for solving the 3C-CNP on general graphs have been proposed in the literature, including greedy approaches [19], [6], [31], integer programming [19], [32], evolutionary algorithms [19], [17], [33], and approximation methods [32], [34]. Considering specific classes of graphs, the 3C-CNP is NP-complete on split graphs and on trees when nodes and connections have nonnegative weights and costs, respectively [20], [35]. The authors of [20] studied the 3C-CNP problem on proper interval graphs and trees. For proper interval graphs, they proposed a polynomial-time algorithm with a time complexity of $O(n^2)$. For trees, they presented an algorithm with a time complexity of $O(n)$ for unweighted trees and $O(n^2)$ for weighted trees. The authors of [36] developed a dynamic programming algorithm for solving the 3C-CNP in polynomial time and space on bipartite permutation graphs. The complexity of their algorithm is $O(nB^2)$. Recently, in [37], the authors presented a polynomial-time algorithm for solving the 3C-CNP on chordal graphs with a maximum node degree of $\Delta = 3$. The proposed algorithm efficiently computes an exact solution with a time complexity of $O(n^2)$. In this paper, we propose a distributed algorithm with a time complexity of $O(n-l)$, where l represents the number of leaf nodes, as detailed in Section III-D.

Table I summarizes the most significant approaches to solving the 3C-CNP problem.

TABLE I
SUMMARY OF THE ALGORITHMIC SOLUTIONS FOR THE 3C-CNP

Citation/ Year	Topology	Complexity class	Solution	Complexity
[30] (2011)	Arbitrary	NP-complete	Centralized	$O(n^2+nm)$
[20] (2016)	Arbitrary with $\Delta \leq 4$	NP-complete	/	/
[20] (2016)	Trees	Linear	Centralized	$O(n)$
[20] (2016)	Weighted trees	Polynomial	Centralized	$O(n^2)$
[20] (2016)	Proper interval	Polynomial	Centralized	$O(n^2)$
[36] (2019)	Bipartite permutation	Polynomial	Centralized	$O(nB^2)$
[35] (2023)	Split	Polynomial	/	/
[37] (2024)	Chordal	Polynomial	Centralized	$O(n^2)$
Proposed	Tree	linear	Distributed	$O(n-l)$

To the best of our knowledge, there are currently no distributed algorithms available for solving 3C-CNP in general or specific graph classes, and no existing studies have applied the 3C-CNP to IoT networks.

III. A Distributed Algorithm for 3C-CNP

In this section, we present the proposed algorithm DA3C-CNP (Distributed Algorithm for the Component-Cardinality-Constrained Critical Node Problem on trees). Before delving into the details of DA3C-CNP, we introduce the variables and functions employed within the algorithm. Subsequently, a detailed description of the algorithm's operation is provided. Finally, we present an illustrative example of its operation.

A. Variables and Primitive Functions

All the main variables and primitive functions used by our algorithm are described in Tables II and III, respectively.

TABLE II
DA3C-CNP VARIABLES DESCRIPTION

Variable	Description
d	Degree of a node
$Critical$	Boolean variable indicating whether a node is critical
S	Cumulative sum of message values received by a node
NR	Number of messages received by a node from its neighboring nodes

TABLE III
DA3C-CNP PRIMITIVE FUNCTIONS

Function	Definition
$send(m, *)$	Broadcasts the message m
$read()$	waits for receipt of messages. This function is blocking if there is no received message any more, it remains blocked in this instruction
$read(wt)$	waits for receipt of messages. If there is no received message after wt milliseconds then the execution will continue and go to the next instruction
$getNNeig()$	returns the number of neighbors of a node
$stop()$	stops the algorithm

B. Description of DA3C-CNP

In this subsection, we provide the pseudocode for DA3C-CNP and explain its execution process in detail. An instance of DA3C-CNP is executed by each node in the network. It operates by taking the upper bound value B as input and producing, for each node, a Boolean output for the variable $Critical$. The algorithm's flowchart is shown in Figure 4. Initially, the DA3C-CNP sets $Critical$ to false for each node, initializes the variable S to 1, and the value of NR (number of received messages) to 1. The degree of the current node is determined using the function $getNNeig()$. The execution begins with leaf nodes, which immediately broadcast a message with value 1 and then terminate their execution. Non-leaf nodes, on the other hand, continue to receive messages from their neighbors. Each time a message is received, the node increments the value of NR and adds the received message values to S. At any point during execution, if the cumulative sum S at a node exceeds the upper bound B, that node is marked as part of the critical nodes set, sets its $Critical$ variable to true, broadcasts a message with value 0, and halts execution. If S remains less than or equal to B, the node waits until all messages from its neighbors are received before proceeding to broadcast its updated S value. This ensures that nodes accurately process the contributions from all neighboring nodes before determining their final status.

Algorithm 1: DA3C-CNP

Data: $B < \frac{n}{2}$
Result: $Critical$
begin
  $Critical \leftarrow false$;
  $S \leftarrow 1$;
  $NR \leftarrow 1$;
  $d \leftarrow getNNeig()$;
  if $(d == 1)$ **then**
    $send(1, *)$;
    $stop()$;
  else
    while *(true)* **do**
      $M \leftarrow read()$;
      $S \leftarrow S + M$;
      $NR \leftarrow SR + 1$;
      if $(S > B)$ **then**
        $Critical \leftarrow true$;
        $send(0, *)$;
        $stop()$;
      else
        if $(NR == d)$ **then**
          $send(S, *)$;
          $stop()$;

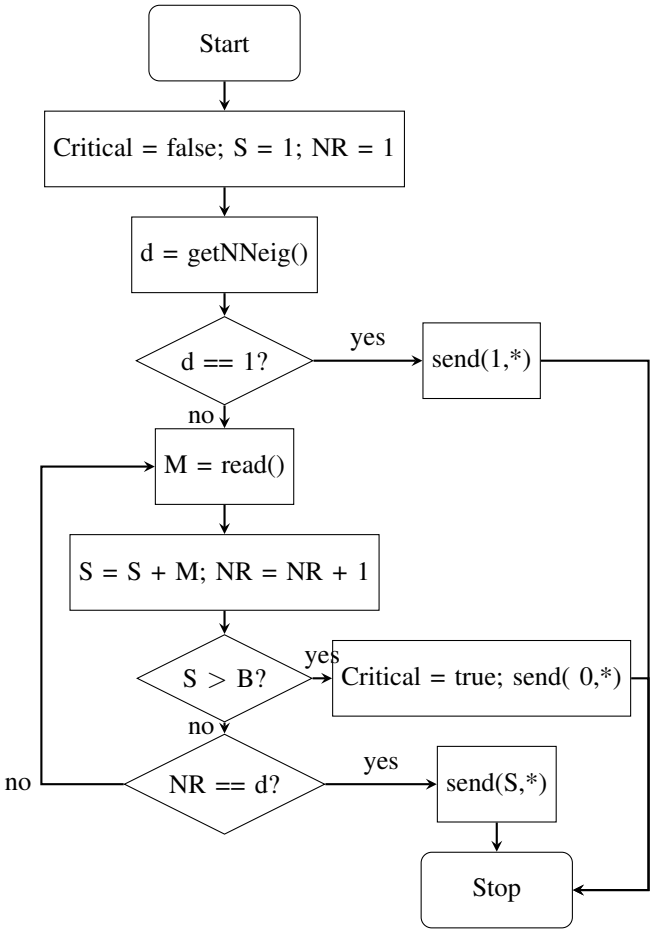


Fig. 4. DA3C-CNP FlowChart

C. Illustrated Example

To illustrate the operation of DA3C-CNP, we apply it to the tree network depicted in Figure 5a. The execution result is

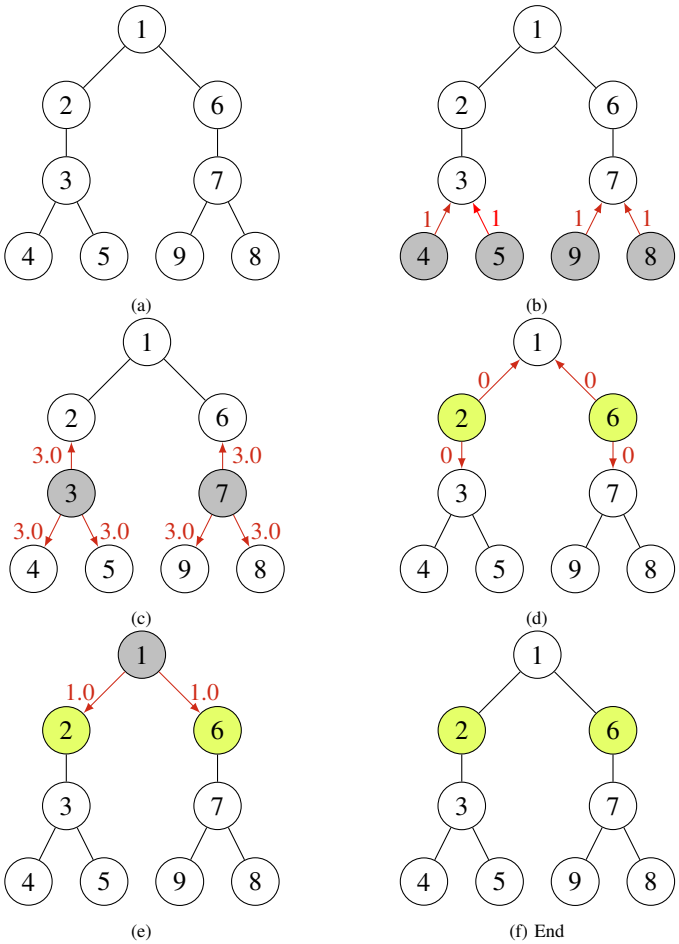


Fig. 5. Illustration of 3C-CNP on a tree IoT network with $B = 3$

illustrated in Figure 5f. In this example, DA3C-CNP operates in five steps: Initially, the leaf nodes (4, 5, 8, and 9) broadcast a message with a value of 0 to their respective parent nodes and terminate the execution of their programs (as shown in Figure 5b). Subsequently, non-leaf node 3 receives two messages, each with a value of 1, from nodes 4 and 5. It sums the received values, yielding $S = 3$. Since the sum is not greater than the upper bound ($B = 3$), node 3 broadcasts a message with a value of 3 (as illustrated in Figure 5c). Similarly, non-leaf node 7 receives two messages with a value of 1 from nodes 8 and 9, respectively, and sums them to $S = 3$. As the sum is not greater than 3 ($B = 3$), node 7 broadcasts a message with a value of 3 (see Figure 5c). Next, nodes 6 and 2 each receive two messages, both with a value of 3. They update their internal variable S, and since the updated values exceed the upper bound B, both nodes 2 and 6 are marked as critical. They broadcast messages with a value of 0 and terminate the execution of their programs (as depicted in Figure 5d). Finally, node 1 receives two messages, each with a value of 0, from nodes 2 and 6. It increments its variable S and halts the execution of its program (see Figure 5e).In summary, the execution of DA3C-CNP on the given network proceeds in five steps, as demonstrated by the progression in Figures 5a through 5f.

D. Complexity

In this subsection, we analyze the complexity of the DA3C-CNP algorithm, focusing on two key metrics: the number of iterations required for execution and the number of messages exchanged during its operation. We establish three main propositions and their corresponding proofs.

Proposition 1. *The DA3C-CNP algorithm identifies a minimal set of critical nodes in a tree-structured IoT network after* $(\Delta - 1)$ *iterations per node.*

Proof. Consider an IoT network modeled as an undirected connected graph $G = (V, E)$ representing a tree, where V is the set of n nodes and E is the set of m edges. Each node $v \in V$ has a degree $d(v)$, satisfying $1 \leq d(v) \leq \Delta$, where Δ is the maximum degree of the graph.

The execution of the DA3C-CNP algorithm begins with the leaf nodes, i.e., nodes with degree $d(v) = 1$. Each leaf node performs a single iteration, during which it sends a message of value 1 to its parent node and then terminates. Thus, leaf nodes require exactly one iteration.

For non-leaf nodes, the execution process is as follows: 1.

1) A non-leaf node waits to receive messages from all its neighbors except its parent node. Upon receiving a message, it increments its message count (NR) and updates its cumulative sum (S) based on the values of the received messages.
2) If at any point the cumulative sum S exceeds the given bound B, the node is identified as a critical node. It broadcasts a message with value 0 and terminates its execution.
3) If $S \leq B$, the node waits until it has received messages from all its neighbors (excluding the parent). Once all messages are received, it sends its cumulative sum S to its parent node and terminates.

In the worst case, each non-leaf node waits for messages from all its child nodes in the tree. Since a node can have at most $\Delta - 1$ child nodes (excluding the parent node in the tree structure), the maximum number of iterations required for a non-leaf node to complete its execution is $\Delta - 1$.

Given the hierarchical nature of tree structures, the execution propagates level by level from leaf nodes to the root node. Thus, the total number of iterations required for the DA3C-CNP algorithm to identify all critical nodes in the tree is bounded by $\Delta - 1$, Δ is the maximum degree of any node in the tree. □

Proposition 2. *In an IoT network, the DA3C-CNP algorithm finds a minimal set of critical nodes using a total of* n *messages.*

Proof. Let the IoT network be represented as a connected tree graph $G = (V, E)$, where V is the set of n nodes and E is the set of edges. Each node $v \in V$ can send at most one message during its execution in the DA3C-CNP algorithm. The proof proceeds as follows:

The tree structure ensures that there are exactly l non-leaf nodes (including the root), and $n - l$ leaf nodes. 1.

1) Leaf nodes, by definition, have a degree of 1. During the execution of the DA3C-CNP algorithm, each leaf node broadcasts a single message to its parent node in the tree structure before terminating execution. Therefore, the total number of messages sent by the leaf nodes is l, where l denotes the number of leaf nodes.
2) Non-leaf nodes wait to receive messages from all their child nodes before taking action. Each non-leaf node then aggregates the received information into a single message, which it forwards to its parent node in the tree. This ensures that each non-leaf node sends exactly one message during its execution. Therefore, the total number of messages sent by the no-leaf nodes is $n - l$.

Since the algorithm involves a single upward communication flow from the leaf nodes toward the root, and every node sends at most one message, the total number of messages exchanged in the network equals the total number of nodes, n. This includes both leaf nodes and non-leaf nodes, as each node participates exactly once in message propagation. □

Proposition 3. *The DA3C-CNP algorithm determines a minimal set of critical nodes in a tree-structured IoT network in fewer than $n - l$ iterations*

Proof. Let the IoT network be represented as a connected tree graph $G = (V, E)$, where V is the set of n nodes and E is the set of edges. Each node $v \in V$ can send at most one message during its execution in the DA3C-CNP algorithm. The proof proceeds as follows:

Each leaf node executes its program independently and in parallel. Upon execution, it sends a single message (value 1) to its parent node and then terminates. This process requires a single iteration for all leaf nodes. However, non-leaf nodes receive messages from all their child nodes. Each time a message is received, the non-leaf node updates its cumulative sum (S) and increments its count of received messages (NR).

The algorithm operates in a level-by-level manner, with execution propagating from the leaf nodes to the root. The total number of iterations is determined by the number of levels in the tree. In the worst case, each level involves the execution of all non-leaf nodes at that level. Given that there are $n - l$ non-leaf nodes in the tree, the total number of iterations required to identify the minimal set of critical nodes is strictly less than $n - l$. □

The DA3C-CNP algorithm offers significant advantages in terms of time complexity compared to existing solutions for the 3C-CNP problem. While the authors of [20] achieved a time complexity of $O(n^2)$ for proper interval graphs and for weighted trees, our algorithm achieves a linear time complexity of $O(n - l)$, making it more efficient for large-scale environments. Similarly, for unweighted trees, although [20] also achieved $O(n)$, our distributed approach provides a scalable and parallelizable solution, which is particularly advantageous in distributed environments. Furthermore, compared to the dynamic programming algorithm proposed in [36] for bipartite permutation graphs, which has a complexity of $O(nB^2)$, our algorithm eliminates the dependency on the parameter B, resulting in a more consistent and predictable performance. Lastly, while the recent work in [37] focuses on chordal graphs with a maximum node degree of $\Delta = 3$, our algorithm's $O(n - l)$ complexity is not restricted by degree of nodes, making it applicable to a broader range of scenarios. Overall, the DA3C-CNP algorithm demonstrates superior efficiency and scalability, addressing the limitations of existing approaches.

IV. Validation and Simulation Results

This section presents the simulation results of the proposed DA3C-CNP algorithm. We implemented the algorithm using the CupCarbon IoT simulator. The primary objective of these experiments is to assess the algorithm's ability to efficiently identify critical nodes within IoT networks while adhering to the specified cardinality constraints.

A. Simulation Tool

To evaluate and validate our algorithm, we employed the widely recognized IoT network simulator, CupCarbon [42]. This simulator, accessible at [43], is highly regarded by researchers, developers, and academics [44], [45]. Its primary objective is to facilitate the design, visualization, debugging, and validation of distributed algorithms, particularly for tasks like monitoring, environmental data collection, and the creation of complex environmental scenarios [45]. It allows for the design and prototyping of networks through an intuitive, user-friendly interface that leverages the OpenStreetMap (OSM) framework to deploy IoT nodes directly onto the map. Each IoT node can be individually configured via a command-line interface using a language specifically designed for CupCarbon, known as SenScript. For more details on the SenScript language, the reader is referred to [45]. The initial version of CupCarbon was introduced by Mehdi et al [46]. Due to its robust capabilities, CupCarbon simulator has gained significant popularity and is widely employed in various research studies. Notable examples of its application can be found in the works of [46], [47], [48].

In the context of graph parameters, the CupCarbon simulator is widely utilized for modeling and analyzing various graph-theoretical constructs. Example parameters include the spanning tree [49], [50], [51], which is fundamental for ensuring efficient network connectivity, and connected components, which identify and classify isolated sub-networks. The simulator also supports the computation of the polygon hull [52], [53], a crucial element in geometric network analysis, as well as leader election [49], [50], [51], [54], an essential problem in distributed systems. Additionally, it facilitates the study of the dominating set [55], [51], which optimizes resource allocation, and the secure dominating set, which introduces robustness and security considerations in network design [56].

B. Experiment

The aim of this subsection is to demonstrate the scalability of the DA3C-CNP distributed algorithm. To achieve this, we utilize the same example presented earlier (see Figure 2). The experiment is conducted in two phases. In the first

phase, the IoT network is designed with a tree topology (see Figure 6) using CupCarbon's intuitive graphical interface, which integrates OpenStreetMap (OSM) for accurate geospatial representation, enabling precise placement of IoT nodes. In IoT networks, critical nodes often serve as essential gateways or aggregation points that bridge multiple devices or network segments. Consider a smart manufacturing facility where hundreds of sensors monitor equipment performance, temperature, and production metrics. Some nodes act as data aggregators that collect information from multiple assembly lines before transmitting it to the central management system.

In the second phase, each node is configured using SenScript, a domain-specific scripting language developed for CupCarbon. SenScript facilitates the detailed modeling of sensor node behavior, including sensing, communication protocols, and energy management. It enables fine control over node actions such as message handling, routing decisions, and power consumption, making it particularly effective for simulating distributed algorithms in large-scale IoT environments. Critical nodes, identified through the DA3C-CNP algorithm, are highlighted in yellow, as depicted in Figure 7.

V. Conclusion

In this paper, we proposed and validated a distributed algorithm, DA3C-CNP, designed to solve the Component-Cardinality-Constrained Critical Node Problem (3C-CNP) in IoT networks. The algorithm identifies critical nodes in tree-structured IoT networks and operates efficiently through local computations and message passing between nodes. A complexity analysis demonstrated that DA3C-CNP converges within $\Delta - 1$ iterations per node, where Δ represents the maximum node degree. Moreover, the algorithm requires the exchange of only n messages, which is optimal for distributed IoT networks. Simulation results, conducted using the CupCarbon simulator, illustrate the effectiveness of the algorithm in a representative IoT network scenario. The algorithm's capability to function in both anonymous and non-anonymous networks underscores its versatility and adaptability, making it suitable for a wide range of IoT applications where reliability and network resilience are critical.

Future work may explore the application of distributed algorithms for the 3C-CNP on other classes of graphs or network topologies, particularly those for which it is tractable. Furthermore, extending the approach to handle large and dynamic networks and integrating it into real-world IoT deployments will be valuable for enhancing communication reliability in critical infrastructure.

References

[1] S. P. Borgatti, "Identifying sets of key players in a social network", Mathematical Organization Theory, Vol. 12, pp. 2134, 2006.

[2] A. Arulselvan, C. W. Commander, L. Elefteriadou and P. M. Pardalos, "Detecting critical nodes in sparse graphs," Computers and Operations Research, Vol. 36(7), pp. 2193–2200, 2009.

[3] C .J. Kuhlman, V. A. Kumar, M. V. Marathe, S. Ravi and D. J. Rosenkrantz, "Finding critical nodes for inhibiting diffusion of complex contagions in social networks," Machine Learning and Knowledge Discovery in Databases, 2010.

[4] D. M. Scott, D. C. Novak, L. Aultman-Hall and F. Guo, "Network robustness index: A new method for identifying critical links and evaluating the performance of transportation networks," Journal of Transport Geography, Vol. 14(3), pp. 215–227, 2006.

[5] M. Guettiche and H. Kheddouci, "Critical links detection in stochastic networks: application to the transport networks," International Journal of Intelligent Computing and Cybernetics, Vol. 12(1), pp. 42–69, 2019.

[6] A. Arulselvan, C. W. Commander, P. M. Pardalos, O. Shylo, "Managing network risk via critical node identification," Risk management in telecommunication networks, pp. 79–92, 2007.

[7] A. Arulselvan, "Network Model for Disaster Management", Ph.D. thesis, University of Florida, 2009.

[8] O. Ugurlu, N. Akram and V. K. Akram, "Critical nodes detection in IoT-based cyber-physical systems: Applications, methods, and challenges". In Emerging trends in IoT and integration with data science, cloud computing, and big data analytics. pp. 226-239, 2022.

[9] Z. Y. Jiang, Y. Zeng, Z. H. Liu and J. F. Ma, "Identifying critical nodes' group in complex networks," Physica A: Statistical Mechanics and its Applications, Vol. 514, pp. 121-132, 2019.

[10] S. M. Senderov and S. V. Vorobev, "Approaches to the identification of critical facilities and critical combinations of facilities in the gas industry in terms of its operability," Reliability Engineering and System Safety, Vol. 203, pp. 107046, 2020.

[11] B. Addis, M. Di Summa and A. Grosso, "Identifying critical nodes in undirected graphs: complexity results and polynomial algorithms for the case of bounded treewidth, ". Discrete Appl. Math, Vol. 161(16), pp. 2349–2360, 2013.

[12] M. Di Summa, A. Grosso and M. Locatelli, "Complexity of the critical node problem over trees," Comput.Oper. Res, Vol 38(12), pp. 1766–1774, 2011.

[13] A. Berger, A. Grigoriev and R. Zwaan, "Complexity and approximability of the k-way vertex cut," Networks, Vol. 63(2), pp. 170–178, 2014.

[14] S. Shen, J.C. Smith and R. Goli, "Exact interdiction models and algorithms for disconnecting networks via node deletions," Discrete Optim, Vol. 9(3), pp. 172–188, 2012.

[15] S. Shen, J.C. Smith, "Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs," Networks, Vol. 60(2), pp. 103–119, 2012.

[16] D.T. Guyen, Y. Shen and M.T. Thai, "Detecting critical nodes in interdependent power networks for vulnerability assessment," IEEE Trans. Smart Grid, Vol. 4(1), pp. 151–159, 2013.

[17] R. Aringhieri, A. Grosso, P. Hosteins and R. Scatamacchia, "A general evolutionary framework for different classes of critical node problems," Eng. Appl. Artif. Intell, Vol. 55, pp. 128–145, 2016.

[18] T.N. Dinh and M.T. Thai, "Network under joint node and link attacks: vulnerability assessment methods and analysis," IEEE/ACM Trans. Netw, Vol. 23(3), pp. 1001–1011, 2015.

[19] A. Arulselvan, C.W. Commander, O. Shylo and P.M. Pardalos, "Cardinality- constrained critical node detection problem," In: In: G¨ulpınar, N., Harrison, P., R¨ustem, B. (eds.) Performance Models and Risk Management in Communications Systems,, pp. 79–91, 2011.

[20] M. Lalou, M.A. Tahraoui and H. Kheddouci, "Component-cardinality-constrained critical node problem in graphs," Discrete Appl. Math, Vol. 210(3), pp. 150–163, 2016.

[21] B. Addis, M. Di Summa and A.s. Grosso, "Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth," Discrete Applied Matehmatics, Vol. 161, pp. 2349–2360, 2013.

[22] Y. Shen, N.P. Nguyen, Y. Xuan and M.T. Thai, "On the discovery of critical links and nodes for assessing network vulnerability," in IEEE/ACM Transactions on Networking, Vol. 21(3), pp. 963–973, 2013.

[23] M. Lalou, M.A. Tahraoui and H. Kheddouci, "The critical node detection problem in networks: A survey," Computer Science Review, Vol. 28, pp. 92–117, 2018. doi:org/10.1016/j.cosrev.2018.02.002.

[24] P. Hosteins, R. Rosario Scatamacchia, A. Grosso and R. Aringhieri, "The connected critical node problem," Theor. Comput. Sci, Vol. 923, pp. 235–255, 2022.

[25] M. Di Summa and S.M.O. Faruk, "Critical node/edge detection problems on trees,"Q J Oper Res, Vol. 40, 2022.

[26] A.Megzari, P.V.P. Raj, W. Osamy and A.M. Khedr, "Applications, challenges, and solutions to single and multi-objective critical node detection problems: a survey," Supercomput, Vol. 79, pp. 19770–19808, 2023.

[27] Z. Niu, Q. Li, C. Ma, H. Li, H. Shan and F. Yang, "Identification of critical nodes for enhanced network defense in MANET-IoT networks," IEEE Access, Vol. 8, pp. 183571-183582, 2020.

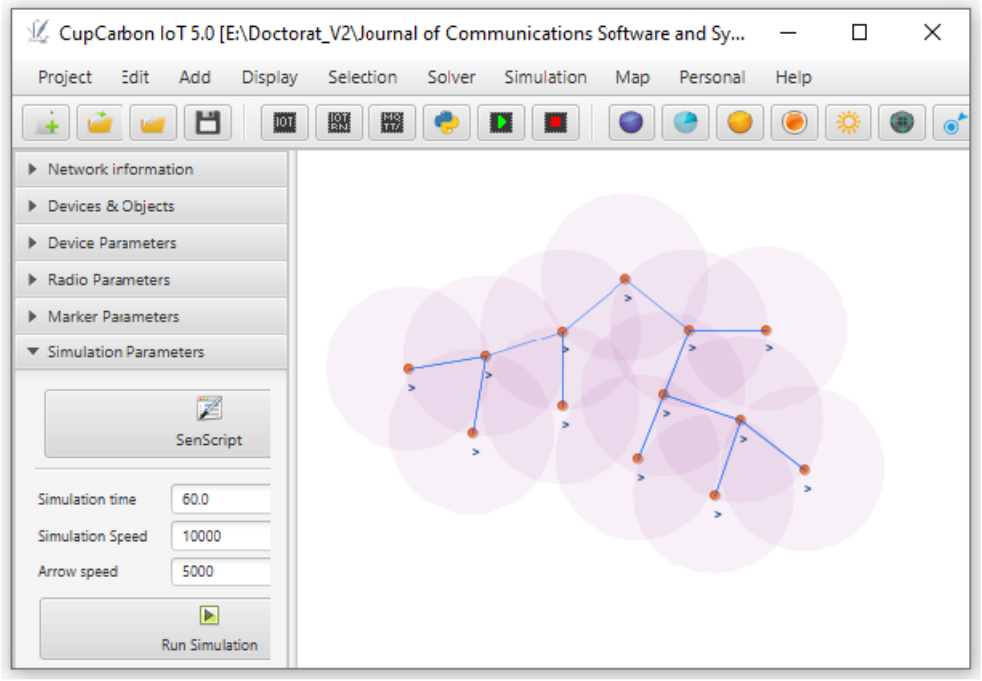


Fig. 6. Design of an IoT Network with 13 Nodes Using the CupCarbon Simulator

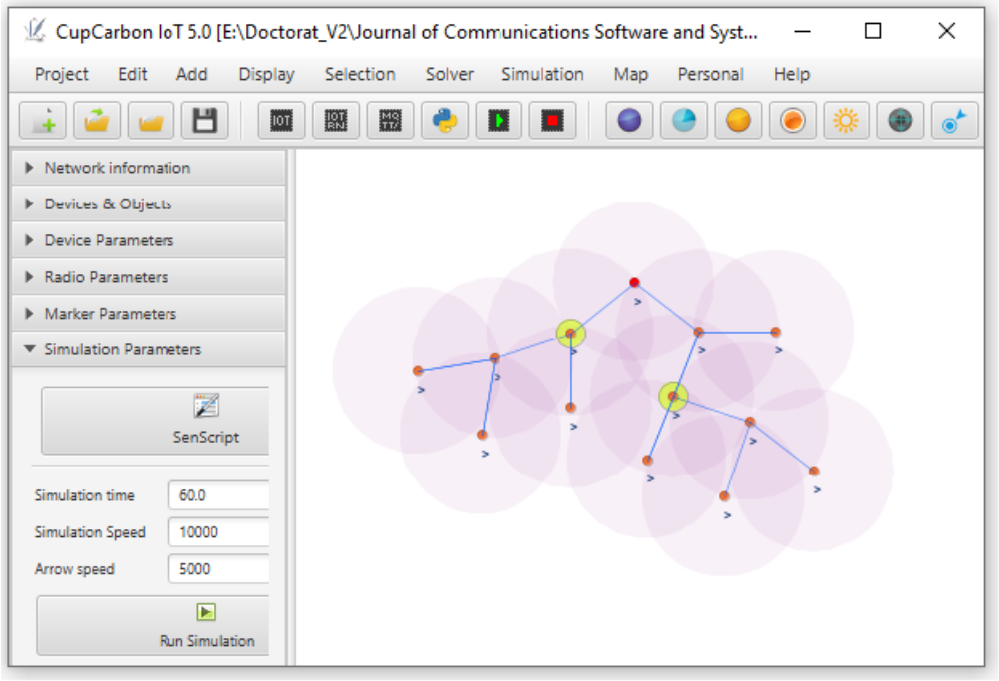


Fig. 7. Example of 2 critical nodes over 13 nodes using CupCarbon IoT simulator.

[28] S. Shukla, "Angle based critical nodes detection (abcnd) for reliable industrial wireless sensor networks," Wireless Personal Communications, Vol 130(2), pp. 757–775, 2023.

[29] I. Ahmad, A. Clark, M. Ali, H. Lei, D. Ferris, A. Aved, "Determining critical nodes in optimal cost attacks on networked infrastructures," Discover Internet of Things, Vol. 4(1), pp. 2, 2024.

[30] A. Arulselvan, C. W. Commander, O. Shylo and P.M. Pardalos, "Cardinality constrained critical node detection problem," Performance models and risk management in communications systems, pp. 79–91, 2011. https://doi.org/10.1007/ 978-1-4419-0534-5 4.

[31] W. Pullan, "Heuristic identification of critical nodes in sparse real-world graphs," J.Heuristics, Vol. 21(5), pp. 577–598, 2015.

[32] E. Balas and C.C. Souza, "The vertex separator problem: a polyhedral investigation," Math. Program, Vol. 103(3), pp. 583–608, 2005.

[33] C. Liu, S. Ge and Y. Zhang, "Identifying the cardinality-constrained critical nodes with a hybrid evolutionary algorithm," Information Sciences, Vol. 642, pp. 119140, 2023. https://doi.org/10.1016/j.ins.2023.119140.

[34] M. Ventresca and D. Aleman, "A randomized algorithm with local search for containment of pandemic disease spread," Comput. Oper. Res, Vol. 48(3), pp. 11–19, 2014.

[35] M. Lalou and H. Kheddouci, "Network vulnerability assessment using critical nodes identification,", In: 2023 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, pp. 1–6, 2023.

[36] M. Lalou and H. Kheddouci, "A polynomial-time algorithm for finding critical nodes in bipartite permutation graphs," Optimization Letters, Vol 13, pp. 1345–1364, 2019. https://doi.org/10.1007/s11590-018-1371-6.
[37] M. Lalou and H. Kheddouci, "Finding important nodes in chordal graphs," In: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE, pp. 1448–1452, 2024. https://doi.org/10.1109/CoDIT62066.2024. 10708414 .
[38] S. Fortunato, "Community detection in graphs," Phys. Rep, Vol. 486(3), pp. 75–174, 2010.
[39] V. Tomaino, A. Arulselvan, P. Veltri and P.M. Pardalos, "Studying connectivity properties in human protein–protein interaction network in cancer pathway," Data Mining for Biomarker Discovery, pp. 187–197, 2012.
[40] M. Lalou and H. Kheddouci, "Least squares method for diffusion source localization in complex networks," In: International Workshop on Complex Networks and Their Applications, pp. 473–485. Springer, 2016.
[41] M. Lalou, H. Kheddouci and S. Hariri, "Identifying the cyber attack origin with partial observation: a linear regression based approach," In: 2017 IEEE 2nd Inter- national Workshops on Foundations and Applications of Self* Systems (FAS* W), pp. 329–333, IEEE, 2017.
[42] A. Bounceur, " Cupcarbon: a new platform for designing and simulating smart-city and IoT wireless sensor networks (sci-wsn)," In: Proceedings of the International Conference on Internet of Things and Cloud Computing(ICC'2016), pp. 1–1, 2016. https://doi.org/10.1145/2896387.2900336
[43] CupCarbon network simulator. Available at: https://cupcarbon.com/.
[44] E. Ojie and E. Pereira, "Simulation tools in internet of things: a review," In: Proceedings of the 1st International Conference on Internet of Things and Machine Learning, pp. 1–7, 2017. https://doi.org/10.1145/3109761.3158400
[45] CupCarbon User Guide. Available at:https://freenwork.com/cupcarbon/cupcarbon user guide.pdf
[46] K. Mehdi, M. Lounis, A. Bounceur and T. Kechadi, "Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool," In: Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques, pp. 126–131. Institute for Computer Science, Social Informatics and Telecommunications Engineering (ICST), Lisbon, Portugal , 2014. https: //doi.org/10.4108/icst.simutools.2014.254811 Security, vol. 421, pp. 75–83. Springer, Singapore, 2022. https://doi.org/10.1007/978-981-19-1142-2 6
[47] D.d.F. Medeiros, C.P.d. Souza, F.B.S.d. Carvalho and W.T.A. Lopes, "Energy- saving routing protocols for smart cities," Energies, Vol. 15(19), pp. 7382, 2022. https: //doi.org/10.3390/en15197382
[48] A. Mukherjee,N. Dey and D. De, "Edgedrone: Qos aware mqtt middleware for mobile edge computing in opportunistic internet of drone things," Computer Communications, Vol. 152, pp. 93–108,2 020. https://doi.org/10.1016/j.comcom.2020.01. 039
[49] A.Bounceur, M. Bezoui, R. Euler, N. Kadjouh and F. Lalem, "Brogo: a new low energy consumption algorithm for leader election in wsns," In: 2017 10Th International Conference on Developments in Esystems Engineering (deSE), pp. 218–223, 2017. https://doi.org/10.1109/DeSE.2017.11
[50] A. Bounceur, M. Bezoui, M. Lounis, R. Euler and C. Teodorov, "A new dominating tree routing algorithm for efficient leader election in IoT networks," In: 15th IEEE Annual Consumer Communications and Networking Conference. CCNC 2018 - 2018, pp. 1–2. Institute of Electrical and Electronics Engineers Inc., United States ,2018. https://doi.org/10.1109/CCNC.2018.8319292
[51] N. Kadjouh, A. Bounceur, M. Bezoui, M.E. Khanouche, R. Euler, M. Hammoudeh, L. Lagadec, S. Jabbar and F. Al-Turjman, "A dominating tree based leader election algorithm for smart cities IoT infrastructure," Mobile Networks and Applications, Vol 28, pp. 1–14, 2023. https://doi.org/10.1007/s11036-020-01599-z
[52] A. Bounceur, M. Bezoui, M. Hammoudeh, L. Lagadec and R. Euler, "Finding the polygon hull of a network without conditions on the starting vertex," Transactions on emerging telecommunications technologies, Vol. 33(3), pp. 3696, 2022.
[53] F. Lalem, A. Bounceur, M. Bezoui, M. Saoudi, R. Euler, T. Kechadi, and M. Sevaux, "Lpcn: Least polar-angle connected node algorithm to find a polygon hull in a connected euclidean graph," Journal of Network and Computer Applications, Vol. 93, pp. 38–50, 2017.
[54] A. Bounceur, M. Bezoui, R. Euler, F. Lalem, "A wait-before-starting algorithm for fast, fault-tolerant and low energy leader election in wsns dedicated to smart- cities and iot," In: 2017 IEEE SENSORS, pp. 1–3, IEEE, 2017.
[55] M. Bezoui, A. Bounceur, R. Euler, F. Lalem, L. Abdelkader, "A new algorithm for finding a dominating set in wireless sensor and IoT networks based on the wait-before-starting concept," In: 2017 IEEE SENSORS, pp. 1–3 ,2017. https: //doi.org/10.1109/ICSENS.2017.8233992 . IEEE
[56] A. Lalouci and F. Zoubeyr, "A distributed algorithm for secure dominating set problem in IoT networks," Studies in Engineering and Exact Sciences, Vol. 5(2), pp. 01–25, 2024. https://doi.org/10.54021/seesv5n2-350
[57] T. Jeyaprakash and R. Mukesh, "An Optimized Node Selection Routing Protocol for Vehicular Ad-hoc Networks – A Hybrid Model," in Journal of Communications Software and Systems, vol. 11, no. 2, pp. 80-85, June 2015, doi: 10.24138/jcomss.v11i2.106
[58] R. Almutairi, G. Bergami, and G. Morgan, "Advancements and Challenges in IoT Simulators: A Comprehensive Review," Sensors, Vol. 24(5), p. 1511, 2024.

Ali Lalouci received his engineering degree in 2009 and his magister's degree in computer science in 2012. He is currently an assistant professor in the Department of Computer Science at Mila University Center, Algeria. He is also a Ph.D. candidate in the Department of Computer Science at the University of Béjaia, Algeria. His research focuses on leveraging distributed computing to enhance the protection of IoT networks.

Zoubeyr Farah received his Ph.D. in Computer Science from the University of Bejaia in 2015. He is currently a faculty member at the University of Bejaia. He is actively involved in research at both the LIMED Laboratory at the University of Bejaia and the LITAN Laboratory at the Higher School of Computer Science and Digital Technologies (ESTIN). His research interests include : Service Composition, Internet of Things (IoT), Cybersecurity, Formal Methods, Artificial Intelligence and its Applications.