Adaptive Monitor Placement for Near Real-time Node Failure Localisation in Wireless Sensor Networks

PAMELA BEZERRA, University of Liverpool, United Kingdom
PO-YU CHEN and JULIE A. MCCANN, Imperial College London
WEIREN YU*, University of Warwick, United Kingdom

As sensor-based networks become more prevalent, scaling to unmanageable numbers or deployed in difficult to reach areas, real-time failure localisation is becoming essential for continued operation. Network tomography, a system and application-independent approach, has been successful in localising complex failures (i.e., observable by end-to-end global analysis) in traditional networks.

Applying network tomography to wireless sensor networks (WSNs), however, is challenging. First, WSN topology changes due to environmental interactions (e.g., interference). Additionally, the selection of devices for running network monitoring processes (monitors) is an NP-hard problem. Monitors observe end-to-end in-network properties to identify failures, with their placement impacting the number of identifiable failures. Since monitoring consumes more in-node resources, it is essential to minimise their number while maintaining network tomography's effectiveness. Unfortunately, state-of-the-art solutions solve this optimisation problem using time-consuming greedy heuristics.

In this paper, we propose two solutions for efficiently applying Network Tomography in WSNs: a graph compression scheme, enabling faster monitor placement by reducing the number of edges in the network; and an adaptive monitor placement algorithm for recovering the monitor placement given topology changes. The experiments show that our solution is at least 1000x faster than the state-of-the-art approaches and efficiently copes with topology variations in large-scale WSNs.

CCS Concepts: • **Computer systems organization → Sensor networks**; • **Networks → Network reliability**; **Network monitoring**.

Additional Key Words and Phrases: Failure Localisation, Network Tomography, Monitors, Dynamic Topologies, end-to-end in-network monitoring

ACM Reference Format:
Pamela Bezerra, Po-yu Chen, Julie A. McCann, and Weiren Yu. 2021. Adaptive Monitor Placement for Near Real-time Node Failure Localisation in Wireless Sensor Networks. *ACM Trans. Sensor Netw.* 1, 1, Article 1 (January 2021), 42 pages. https://doi.org/10.1145/3466639

1 INTRODUCTION

There is an increased proliferation of sensing and control systems based on networks of wireless sensors (WSNs), such as in Smart Cities [41, 56], Industry 4.0 [1, 30], and Precision Agriculture

*Weiren Yu was supported by the National Natural Science Foundation of China (61972203) and Natural Science Foundation of Jiangsu Province (BK20190442)

Authors' addresses: Pamela Bezerra, pamelab@liverpool.ac.uk, University of Liverpool, Liverpool, United Kingdom; Po-yu Chen, po-yu.chen11@imperial.ac.uk; Julie A. McCann, j.mccann11@imperial.ac.uk, Imperial College London, London, United Kingdom; Weiren Yu, weiren.yu@warwick.ac.uk, University of Warwick, Warwick, United Kingdom.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
1550-4859/2021/1-ART1 $15.00
https://doi.org/10.1145/3466639

[2, 33]. As uptake of such systems is becoming more prevalent, they have the potential to scale to hundreds or thousands of potentially heterogeneous devices, with each device being typically resource-constrained and low-powered and capable of communicating via wireless channels. These constraints together with constant cyber-physical interactions with the real world, mean that these devices are commonly fault-prone and vulnerable to malicious attacks, natural damage, and errors. Given this, malfunctioning devices (i.e., *node failures*) can result in data loss or corruption, communication delays, or network partitioning.

Additionally, sensor-based devices are being embedded into a variety of artefacts (e.g. pumps [31]) and infrastructures (e.g. roads [23]), with some being deployed either in places difficult to reach (e.g. bridges [60]) or in hazardous environs (e.g. power plants [61]), where collecting them to re-calibrate, replace, or fix is potentially expensive and/or dangerous. This issue is exacerbated as such networks scale and the devices have to serve both sensing and communications functions (e.g. in multi-hop networks). To avoid the resulting error propagation therein, and to maintain long-term WSN functionality, it is becoming more essential to efficiently and accurately identify node failures.

The design of node failure localisation solutions for WSNs, however, is extremely challenging due to the aforementioned devices' resource constraints and the growing scale of these systems. Therefore, a large number of approaches have been recently proposed to solve this challenge [48, 55, 74] (further details in Section 2). Among these state-of-the-art solutions, *network tomography* is a non-intrusive and flexible approach capable of accurately localising node failures in different WSN topologies and applications while incurring relatively minimal additional overheads, or costs, to the monitored network [19, 26, 32, 59].

Network Tomography localises node failures based on end-to-end path measurements performed by a subset of devices called *monitors*. These monitors, which can either be *soft-monitors* (standard devices with software to carry out network monitoring tasks [32, 36]) or *hard-monitors* (additional dedicated hardware [14, 65]), are responsible for collecting and analysing different in-network metrics, such as packet delivery ratio and delays, from multiple communication paths by exchanging *data probes* among themselves. Given that a path is considered abnormal if at least one device in this path is acting abnormally, Network Tomography is capable of generating a system of linear equations describing the state of every device traversed by probes, as demonstrated in Fig. 1. By solving this system, it is possible to obtain the number k ($k \geq 1$) of simultaneous node failures that a given set of monitors, distributed around the network, is capable of localising. In this paper, we focus on *soft-monitor based Network Tomography* due to its easy deployment and agility to cope with the topology variations common to WSNs.

Since Network Tomography depends only on the end-to-end performance of different measurement paths, it is a *system and application-independent approach*, requiring neither a priori knowledge of the network nor the devices' having to communicate status reports to localise node failures. This design enables the system to accurately localise node failures even in a hostile environment (where many nodes are malicious or malfunctioning) as well as more complex failures, such as hidden failures (i.e., failures that look like normal behaviour from the local node view [59]) or silent failures (hitherto unknown failures caused by unanticipated interactions between the network and the environment [52]). Additionally, the probing scheme (i.e., the way the probes move between monitors) can be easily implemented by exploiting the system's existing communication protocols, either for data routing (e.g, RPL [50]) or network management (e.g., polite flooding like Glossy [53]), which reduces implementation costs and communications overhead in the failure localisation process. Further details on Network Tomography are presented in Sections 2 and 3.

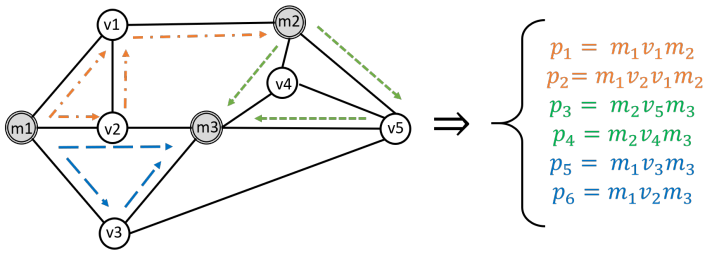


Fig. 1. Example of a small network $\mathcal{G}$ with three monitors (m_1, m_2, m_3) and five non-monitors (v_1, v_2, v_3, v_4, v_5), and some examples of measurement paths used by the probes (dashed lines). The linear equation system on the right describes the nodes observed by each path (equations have the same colour of the corresponding paths in). By checking the behaviour of the probes crossing these paths (e.g., delay) and solving the system of equations, monitors are capable of uniquely identifying a certain number node failures. The higher the number of equations, the higher the number of nodes observable and, therefore, the capability of the network in localising simultaneous node failures. The system of equations, however, is impacted by the number and position of the monitors.

The selection of devices to act as soft-monitors, a process called *Monitor Placement*, is, however, a non-trivial task. Soft-monitors consist of a subset of deployed devices uploaded with the necessary algorithms for reliable failure localization. Since they are capable of collecting end-to-end measurement data for long periods of time (details on Section 3), they consume more energy compared to the remaining devices (also known as *non-monitors*), and it is desirable to reduce their number. On the other hand, the higher the number of soft-monitors, the higher the number of uniquely identifiable node failures (described by the value of k), making the network more robust against multiple simultaneous node failures. Therefore, the monitor placement problem can be formulated as an optimisation problem aiming to minimise the number of soft-monitors given a desirable threshold of k. The solution to this optimisation problem has been proven to be NP-hard [11, 64] and is typically solved in polynomial time with greedy heuristics. Among these heuristics, *maximum node-identifiability monitor placement* (MNMP) [43] has shown great potential for finding sub-optimal monitor placement for localising node failures in generic topologies.

Although MNMP provides a feasible solution for the Monitor Placement problem and is frequently used in fixed topology systems (such as IP Networks), its high computational overhead prevents it from being applied to WSNs. Our initial experiments on a high-performance computer (details in Section 3) show that, given a small network consisting of only 318 nodes and 758 edges, MNMP required 1.5 hours to obtain the monitor placement for localising as little as two simultaneous node failures ($k = 2$). This low performance is due to the greedy nature of MNMP, which tests all possible combinations of monitor sets that maximises the current value k. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, MNMP selects new monitors by computing the node identifiability index $\Omega(n)$ of *every* non-monitor $n \in \mathcal{N}$ ($\mathcal{N} \subset \mathcal{V}$). This index is relevant for measuring the number of node failures k a monitor set can localise and it is computed using the maximum-flow algorithm [25] (details in Section 3). Given this, the time complexity for selecting a new monitor with MNMP is given by $O(|\mathcal{E}|^2|\mathcal{N}|^3)$. Our experiments show that given two networks with 210 and 604 nodes, MNMP requires 24 and 752 minutes, respectively, to obtain the best monitor placement for k = 2.

It is clear from this analysis that a MNMP based heuristic cannot efficiently select the best soft-monitor set for WSN since it cannot cope with (1) the discussed growing scale of these systems and (2) the commonly observed variations in topology, which are usually caused by interference [7] and/or devices battery depletion. Given that changes in topology lead to changes in the measurement

paths and, consequently, the system of equations (Fig. 1), MNMP cannot efficiently adapt the monitor placement, reducing the accuracy of the node failure localisation process over time.

In this paper, we aim to solve the monitor placement problem for large-scale WSN from two different perspectives. Firstly, given that the most time consuming step in MNMP is computing the maximum-flow between every non-monitor and the monitor set $\mathcal{M}$, can we reduce the execution time by reducing the number of edges $|\mathcal{E}|$ or non-monitors $|\mathcal{N}|$ of a given network $\mathcal{G}$? Secondly, given a set of monitors $\mathcal{M}$ and changes in topology (i.e., a list of failed communication links or devices), can we minimise unnecessary operations and efficiently adapt $\mathcal{M}$ to maintain the node failure localisation accuracy? To answer these two questions, we propose two new algorithms, namely a **graph compression scheme** and an **adaptive monitor placement (AMP) algorithm**. These contributions are summarised below:

- We propose a scalable graph compression algorithm that can *efficiently aggregate the number of edges with similar neighbouring topologies*, transforming an original graph $\mathcal{G}$ into a compressed graph $\hat{\mathcal{G}}$. Our technique can effectively reduce the number of edges for each biclique (complete bipartite subgraph) $(\mathcal{X}, \mathcal{Y})$ in $\mathcal{G}$ from $|\mathcal{X}| \times |\mathcal{Y}|$ to $|\mathcal{X}| + |\mathcal{Y}|$. Users can use this technique to **compress networks before computing initial monitor set, reducing the computational time of both MNMP and AMP**. To the best of our knowledge, we are the first to successfully apply this edge aggregation technique to the maximum-flow problem in large-scale networks, efficiently reducing the number of edges while accurately computing the network's maximum-flow.
- We propose an adaptive monitor placement (AMP) algorithm, which can quickly identify the nodes affected by topology changes, given by the set $\mathcal{A} \subseteq \mathcal{N}$. We further propose a novel way to reduce the set of potential candidate monitors from $\mathcal{N}$ to $\mathcal{C} \subseteq \mathcal{N}$ by formulating the candidate selection problem as a set cover problem (SCP). Both processes **reduces the search space for new monitors, making the greedy heuristic more intelligent under topology changes**. Comparing with MNMP, which yields $O(|\mathcal{E}|^2|\mathcal{N}|^3)$ computational overhead, AMP reduces computational overhead to $O(|\mathcal{E}|^2|\mathcal{N}||\mathcal{A}||\mathcal{C}|)$. Since $|\mathcal{A}| \ll |\mathcal{N}|$ and $|\mathcal{C}| \ll |\mathcal{N}|$ (which is proven by our experiments in Section 7), AMP is significantly faster then MNMP, especially in large networks. Consequently, AMP can cope with the topology changes that are frequently observed in WSNs, keeping the network capable of accurately localising multiple node failures even under frequent variations in topology.
- We compare our solution against the state-of-the-art approaches in both real-world and synthetic networks. Our experiments show that, with our graph compression and AMP algorithm, the network identifiability recovery time is **1,000x faster than MNMP for the majority of the networks studied considering a failure scenario where 10% of the nodes fail** (i.e., are removed for the network). This shows that our approach is capable of efficiently adapting the monitor placement to the dynamic topology changes common to static WSN. More details on the experiment setups, the network studied and failure models explored are described in Section 7.
- Apart from extensive experiments, we also explore a case study based on the London bus-stop network (Section 8), a smart city application with more than 1000 nodes spread around the city of London-UK. We demonstrate that, given a topology change where 10% of the communication links fail, our solution could recover the network identifiability index within 250 seconds in a large network with 3,804 nodes and 133,111 links. A comprehensive analysis on the size of the candidate monitor set $\mathcal{C}$ in relation to the affected set $\mathcal{A}$ based on different topology changes and network properties is also presented in this case study.

The remainder of this paper is organised as follows: a brief review of the state-of-the-art in node failure localisation and network tomography is presented in Section 2. The necessary background for understanding this work is described in Section 3. The problem statement and a solution overview are given in Section 4. Our graph compression and adaptive monitor placement (AMP) algorithms are presented in Sections 5 and 6, respectively. Evaluation and a case study based on London bus-stop networks are given in Sections 7 and 8, respectively. Finally, we conclude this work in Section 9.

2 RELATED WORK

2.1 Failure Localisation in WSNs

The literature in node failure localisation for WSNs can be mainly classified into **Centralised**, **Distributed**, and **Hybrid solutions** [48, 55, 74].

Centralised approaches use the Base Station (BS) as an arbiter, processing monitoring data and spreading diagnosis messages across the network [38, 66, 71]. It is one of the most accurate node failure localisation methods since the BS has a global view of the network and more resources compared to the nodes. However, the high amount of monitoring data transmitted between nodes and the BS incurs in high network congestion and latency, making near real-time failure localisation difficult.

In **distributed approaches**, each node is responsible for localising failures in its one-hop neighbourhood by comparing its current status data with its neighbours' to achieve consensus [5, 6, 40, 67, 72]. Communication overhead and delay are reduced since more decisions are made locally. However, these approaches are expensive in terms of data processing, memory, and energy, since each node performs monitoring processes in addition to the usual systems tasks. Moreover, since the localisation process is highly dependent on the neighbouring nodes report (i.e., each node communicates its status - failed or normal - to its neighbours) the accuracy of distributed approaches degrades in a hostile environment, where many nodes are either malfunctioning or malicious and, therefore, transmitting inaccurate information.

Finally, distributed approaches are incapable of detecting more complex failures, such as hidden failures [59], since they only have a local-view of individual nodes. An example of hidden failure is a Sinkhole attack [39], where each node keeps transmitting data correctly, but to a malicious base station (BS). To accurately localise this malicious BS, it is necessary to observe the *paths used by the data packets* instead of the node's behaviour.

Hybrid approaches have been proposed recently to reduce the high latency and network congestion of centralised solutions while avoiding the node processing and memory overhead common to distributed approaches. In these approaches, a set of dedicated devices, either cluster-heads (CH) (cluster-based solutions) or monitors (hardware or software monitor-based solutions), are allocated to perform network analysis. **Cluster-based** solutions implement clustering protocols for node failure localisation, assigning different CHs to monitor the local network inside each cluster [13, 47]. Each CH is responsible for processing and aggregating data from the monitored nodes, communicating with the BS when necessary. This approach, however, can be efficiently applied only in networks using clustering protocols for data routing, since clusters disrupt the network topology and degrade the routing process. Additionally, the CH still depends on each node report to localise failures, suffering from similar accuracy limitations of distributed solutions.

Meanwhile, **Monitor-based approaches** employ **monitors** distributed over the network to localise failures. Since these monitors are only responsible for node failure localisation, monitor-based approaches can be applied to different systems and routing protocols, detaching the data communication from the monitoring process and, consequently, reducing bandwidth consumption.

Monitors can be either a subset of standard devices uploaded with network monitoring algorithms, which we defined as **soft-monitors** [32, 36], or additional and powerful hardware, such as drones or gateways, also known as **hard-monitors** [14, 65]. Soft-monitors solutions are easier and faster to deploy, as well as capable of adapting to changes in the monitored system, since it only require the upload of monitoring algorithms to a subset of nodes in strategic positions. Nodes selected to become monitors switch their behaviour from data collection and transmission to network monitoring and failure localisation. Given this, this paper focus on the *soft-monitor placement problem*, and how to efficiently adapt this placement to changes in the network topology.

2.2 Network Tomography

Among the soft monitor-based approaches, **Network Tomography**, a method for inferring network performance based on end-to-end measurements, has recently gained the attention of the WSN community [20, 27, 32, 59, 75]. In this approach, monitors exchange *probes* among themselves through measurement paths to collect network metrics and identify anomalous links. An example of a typical network metric observed is delay (also known as Delay tomography [26, 36]), where monitors send multiple probes through different paths and compare their travel times.

Assuming that a communication path is considered normal only when all nodes along it behave normally, monitors can localise node failures by identifying (1) which measurement paths are abnormal and (2) which nodes are common to these paths. This problem, known as the *identifiability problem*, consists of a system of linear algebra equations, where the behaviour of the nodes (normal or failure) represent the variables, and the state of the measurement paths are the constants. Figure 1 presented in the Introduction details an example of these equations.

Given the above, it is possible to observe that Network Tomography is independent of the network's routing protocol or application domain, with its efficiency being impacted only by the number and position of monitors. Additionally, these monitors can transmit probes through any available network layer, either for data routing [32, 50] or for network management (e.g., polite flooding protocols [36, 46]). These unique characteristics not only reduce the development and deployment costs necessary for implementing failure localisation, but also minimises the monitors' communication overhead.

Apart from that, the end-to-end analysis of different communication paths enables Network Tomography to localise more complex failures, such as Hidden or Silent Failures. As previously mentioned, Hidden Failures behaves like a normal device based on local analysis. Given the discussed Sinkhole attack, Network Tomography is capable of localising the malicious BS by analysing, for example, the packet loss of different measurement paths traversing the real BS. Meanwhile, Silent Failures [52] are failures caused by unpredicted interactions between the nodes and the environment. Since these failures are not known at development time, neither built-in monitoring mechanisms nor application-specific localisation schemes are capable of identifying them. Network Tomography, however, is capable of identifying any problematic node based on the analysis of different network metrics from multiple paths.

2.3 Monitor Placement

As previously explained, the number and position of monitors define the node failure localisation capabilities of the network (i.e., its reliability). This capability is described by the number $k \geq 1$ of *uniquely identifiable node failures* a given monitor set $\mathcal{M}$ can localise, with higher k requiring a higher number of monitors. On the other hand, given that monitors are reliable and secure devices that don't fail during the monitoring process (details on Section 3.2), it is desirable to minimise the set of monitors due to the WSN resource-constraints. Additionally, a smaller number of monitors constantly transmitting probes results in less bandwidth consumption.

The **monitor placement** problem is an optimisation problem for minimising the monitor set $\mathcal{M}$ given a desired value of k, which is defined by the application reliability requirements. This optimisation problem is, however, proven to be NP-hard [11, 64], with many proposed solutions imposing constraints on the type of topology (e.g., tree) or the maximum number of monitors (e.g., $|\mathcal{M}| = 2$) to solve it efficiently [3, 15].

A set of sub-optimal solutions for generic network topology were proposed in [43, 45]. Maximum Node-identifiability Monitor Placement (MNMP) is a greedy heuristic that gradually adds and removes monitors to the monitor set $\mathcal{M}$ to find the optimum combination that maximises the network identifiability and achieve k. Recent work has proposed extensions to this algorithm [9, 44, 73], some of which are specific for certain types of topologies [9] or application scenarios [73]. However, none of these approaches are suitable for WSN since they do not consider variations in network topology, which impacts the measurement paths used by the probes, and the importance of constantly adjusting the monitor set to maintain the system's reliability over time.

In this paper, we propose monitor placement solutions which can quickly react to changes in topology common to WSN. For better understanding the challenges in adjusting the monitor placement, in Section 3 we detail some concepts relevant to Network Tomography along with our network model and notations.

3 BACKGROUND

3.1 Network Model

We define our system as a *multi-hop mesh WSN* modelled as a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of nodes (e.g., devices) and edges (e.g., wireless communications links), respectively. Here we define every communication link in $\mathcal{E}$ as a Boolean value reflecting the connectivity of such a link, with 1 meaning a connection exists and 0 otherwise. Under this model, network tomography is known as *Boolean Network Tomography* [27]. This binary description of the communication links simplifies the mathematical assumptions necessary to prove the identifiability conditions (detailed in Subsection 3.4).

A subset of nodes $\mathcal{M} \subseteq \mathcal{V}$ is defined as **monitors** (Subsection 3.2). These nodes are responsible for monitoring tasks only, measuring network characteristics by sending probes along certain paths defined by the network topology and the probing schemes (Subsection 3.3). The remaining nodes are defined as **non-monitors** $\mathcal{N} = \{\mathcal{V} \setminus \mathcal{M}\}$. These nodes carry out the usual system tasks (i.e., data sampling, processing, and transmission) and can fail (i.e., act abnormally) once or more times during the system's lifetime. We consider that multiple simultaneous node failures might occur given a time interval t. For simplicity, we assume these node failures to be *independent*.

3.2 Monitors

In this paper we assume that monitors are secure devices with enough available energy, providing reliable network analysis. When a node is selected as a monitor, the Base Station (BS) uploads it with the necessary algorithms for monitoring the network, together with the identity of the remaining monitors. The monitor set $\mathcal{M}$ is, therefore, known only by the BS and the monitors, and cryptography keys could be shared among them for achieving higher security [36]. After the code update, monitors coordinate among themselves and the BS to identify and store all the available measurement paths and start the failure localisation process.

To reduce energy consumption and communication overhead, we also assume that monitors uses the network's available communication protocols for transmitting probes, instead of implementing additional protocols (details in Section 3.3).

3.3 Probing Schemes

Probing schemes are the protocols used by monitors to send and receive probes. These schemes can be mainly classified into Controllable Arbitrary-path Probing (CAP), Controllable Simple-path Probing (CSP), and Uncontrollable Probing (UP) depending on the types of measurement paths used [43–45]. CAP and CSP are two common probing schemes that are independent of routing protocols. While CAP allows probes to travel via any possible path, either simple or cycle, in CSP probes travel only via simple paths between two distinct monitors. In contrast, UP allows probes to travel via *only* the set of paths defined by the routing protocol adopted.

In this paper, we focus on CAP and CSP schemes since fault-management systems typically have higher priority than data streams. Additionally, these probing schemes can use all wireless links between nodes, even if these links are not used for data routing, enabling us to develop generic, routing protocol-agnostic monitor placement solutions.

Apart from this, a number of lightweight and efficient CAP and CSP schemes have been recently proposed for WSN Network Tomography [26, 46, 59, 75]. The authors of [46] and [36], for example, implement a CAP scheme by using the flooding protocols employed to perform network management processes (e.g., synchronisation and software update) to transmit probes. Similarly, many WSN use polite flooding protocols, such as Trickle [28] and Glossy [53], or controlled broadcasting (e.g., EXT [17]) for network management, which can be adapted for collecting network metrics. For instance, considering a RPL network [34, 50], monitoring data could be included in the RPL DIO messages sent downstream to the nodes for routing tree construction and the DAO messages sent back upstream for node rank advertisement.

Considering that both the monitor set $\mathcal{M}$ and the probing schemes are well defined, the *Maximum Network Identifiability Index* $\Omega(\mathcal{N})$ measures the number of simultaneous node failures the analysed network $\mathcal{G}$ can uniquely identify. Since this index is extremely relevant for the monitor placement problem, in the following sections (Sections 3.4, 3.5, and 3.6), we will detail the concept of Network Identifiability and explain how to compute $\Omega(\mathcal{N})$.

3.4 Network Identifiability

Boolean Network Tomography can uniquely identify an individual node failure $f \in \mathcal{F}$ within a non-monitor set $\mathcal{N}$ when a network is *identifiable* as defined below:

Definition 1 (identifiable). *Let a failure set $\mathcal{F} \subseteq \mathcal{N}$ be a set of non-monitors that fail simultaneously, and $\mathcal{P}_{\mathcal{F}}$ contains all the failed communication paths that are caused by node failures in $\mathcal{F}$. Two failure sets $\mathcal{F}_1$ and $\mathcal{F}_2$ are considered* identifiable *(i.e., distinguishable) if and only if their failed paths are non-identical (i.e. $\mathcal{P}_{\mathcal{F}_1} \neq \mathcal{P}_{\mathcal{F}_2}$) with respect to $\mathcal{G}$ [45]. This means that there is a path that traverses at least one node in one set and none of the nodes in the other set.*

From Definition 1, it can be easily observed that, when the sizes of two failure sets grow, it is less likely that these failures are identifiable since there is a higher chance that the failed paths will overlap. In order to describe how many failures a network can tolerate, while correctly identifying distinct failures, the maximum network identifiability index, $\Omega(\mathcal{N})$ is given by Definition 2 below:

Definition 2 (maximum identifiability index $\Omega(\mathcal{N})$). *Given a network $\mathcal{G}$ with non-monitor set $\mathcal{N}$ [44, 45]:*

1. *$\mathcal{N}$ is k-identifiable if for any two distinct failure sets $\mathcal{F}_1$ and $\mathcal{F}_2$ ($\mathcal{F}_1 \neq \mathcal{F}_2$) are identifiable when $|\mathcal{F}_i| \leq k$ ($i = 1, 2$).*
2. *The maximum network identifiability index of $\mathcal{N}$, $\Omega(\mathcal{N})$, is the maximum value of k such that $\mathcal{G}$ is k-identifiable.*

In summary, the maximum network identifiability index $\Omega(\mathcal{N})$ measures the network's capability of uniquely localising $\Omega(\mathcal{N})$ numbers of simultaneous node failures using Boolean Network Tomography given a set of non-monitors $\mathcal{N}$. It is worth noticing that the identifiability definitions above can be applied to any subset of $\mathcal{N}$, including the unitary subset $S = \{n\}$. Given this, it is possible to characterise identifiability in terms of *per-node property* according to the following definition [44]:

Definition 3. *Given a network $\mathcal{G}$ with a set of non-monitors $\mathcal{N}$:*

(1) *If $\mathcal{N}$ is k-identifiable, then any $n \in \mathcal{N}$ must be k-identifiable.*
(2) *If n is k-identifiable $\forall n \in \mathcal{N}$, then $\mathcal{N}$ is k-identifiable.*
(3) *The maximum network identifiability index $\Omega(\mathcal{N})$ can be defined in terms of the node identifiability index $\Omega(n)$ as $\Omega(\mathcal{N}) = min_{n \in \mathcal{N}} \Omega(n)$*

3.5 Auxiliary Graph Transformation

Considering the definitions in Section 3.4, the maximum network identifiability index $\Omega(\mathcal{N})$ is obtained by finding the number of unique measurements paths $p \in \mathcal{P}$ given $\mathcal{M}$ and $\mathcal{G}$. The current state-of-the-art proposes the use of *auxiliary graphs* $\mathcal{G}^*$ to solve this problem at polynomial time [43–45]. This graph is generated by replacing all monitors in the monitor set $\mathcal{M}$ with a single monitor m', called *virtual monitor*, and connecting all the neighbour's nodes of $\mathcal{M}$, $N(\mathcal{M})$, to this new monitor. This transformation is formalised by the equation below [44]:

$$\mathcal{G}^* := \mathcal{G} - \mathcal{M} + \{m'\} + \mathcal{L}(\{m'\}, N(\mathcal{M})) \tag{1}$$

Figure 2 illustrates this process. The auxiliary graph $\mathcal{G}^*$ describes, therefore, the level of connectivity between non-monitors and the virtual monitor m', and $\Omega(\mathcal{N})$ can be obtained by solving the minimum-cut problem in $\mathcal{G}^*$ [18] (details in Subsection 3.6).

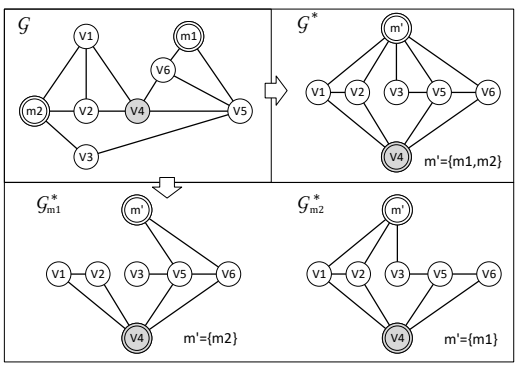


Fig. 2. An example of a network $\mathcal{G}$ with two monitors, m_1 and m_2, and the obtained auxiliary graphs for CAP ($\mathcal{G}^*$ on top) and CSP ($\mathcal{G}^*_{m1}$ and $\mathcal{G}^*_{m2}$ at the bottom). For CAP, m_1 and m_2 are replaced with a virtual monitor m' (i.e. $m' = \{m_1, m_2\}$). CSP require extra auxiliary graphs since it only allow probes to traverse simple paths starting and finishing at different monitors. Two more auxiliary graphs $\mathcal{G}^*_{m1}$ and $\mathcal{G}^*_{m2}$ are generated to represent the combination of paths starting and/or finishing in m_1 or m_2

It is worth mentioning that different probing schemes require different auxiliary graphs. For CAP, one graph is sufficient to compute $\Omega(\mathcal{N})$ since probes can follow any paths. CSP, however,

requires extra graph transformations since the probes can only follow simple paths that begin and end at different monitors. A number $|\mathcal{M}|$ of extra auxiliary graphs are, therefore, generated in addition to the original one $\mathcal{G}^*$ to represent different combinations of pairs of monitors [44]. For each monitor $m \in \mathcal{M}$, an auxiliary graph $\mathcal{G}^*_m$ is generated by replacing all monitors except m by the virtual monitor m'. Consequently, each $\mathcal{G}^*_m$ is defined as [44]:

$$\mathcal{G}^*_m \coloneqq \mathcal{G} - \mathcal{M} + \{m'\} + \mathcal{L}(\{m'\}, N(\mathcal{M}\backslash\{m\})) \tag{2}$$

The value of $\Omega(\mathcal{N})$ obtained for CSP is, therefore, the minimum value computed from all auxiliary graphs.

3.6 The maximum flow minimum cut problem

With the auxiliary graph $\mathcal{G}^*$, it is possible to compute $\Omega(\mathcal{N})$ in polynomial time by finding the minimum cut of $\mathcal{G}^*$. The minimum cut of a connected graph is the minimum set of edges whose deletion disconnects a graph [18]. In the case of the auxiliary graphs, the minimum cut gives us the minimum set of measurement paths connecting non-monitors and the virtual monitor m' (i.e., the monitor set $\mathcal{M}$), resulting in the value of $\Omega(\mathcal{N})$.

According to the maximum-flow minimum-cut theorem [18, 25], the minimum cut of a connected graph can be obtained trivially by computing the maximum flow of the same graph. Given a weighted graph, the *flow* of a path is the amount of commodity that can be moved from a source node s to a sink node t, with the edges' weight representing the maximum capacity of that path. The maximum flow problem consists of finding the maximum flow f_m between s and t. The first known algorithm for solving this problem, named Ford-Fulkerson, was proposed in [25] and consists of searching for *augmenting paths*, paths with available capacity between source and sink, until no more paths are available. The most common implementation of this algorithm uses the breadth-first search (BFS) to find the augmenting paths [21], which requires $O(|\mathcal{E}^2||\mathcal{V}|)$ the time complexity, where $|\mathcal{V}|$ and $|\mathcal{E}|$ are the numbers of vertices and edges of the connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, respectively.

3.7 Monitor Placement Problem and MNMP Algorithm

As discussed in Section 3.2, it is assumed that monitors are secure and reliable entities that do not fail during the failure localisation process. In order to reduce resource and bandwidth consumption, it is desirable to *minimise* the number of monitors necessary to guarantee that $\Omega(\mathcal{N})$ is above a desirable identifiability level k, which can be formally defined as the optimisation problem below:

$$\begin{aligned} \underset{\mathcal{M}}{\text{minimize}} \quad & |\mathcal{M}| \\ \text{subject to} \quad & \Omega(\mathcal{N}) \geq k,\ \mathcal{N} = \{\mathcal{V} \setminus \mathcal{M}\} \end{aligned} \tag{3}$$

where k is a constant defined by the systems' reliability requirements.

This optimum monitor placement problem has been proven NP-hard and is typically solved with greedy heuristic-based solutions [11, 64] (discussed in Section 2). The state-of-the-art approach effectively solves this problem in polynomial time using the maximum node-identifiability monitor placement (MNMP) algorithm [43], which reduces the original problem into a maximum flow minimum cut problem in its corresponding auxiliary graphs according to the probing scheme adopted (as explained in Sections 3.3, 3.5, and 3.6).

This algorithm can be regarded as a greedy solution consisting of the following steps:

- **Step 1 - Ensuring Coverage**: In the first step, MNMP selects a set of monitors that ensures that all non-monitors are covered by at least one measurement path. Under CAP, a single monitor is randomly selected yielding $O(1)$ time overhead. For CSP, it is necessary to search

for the 2-connected components and their cut vertices to find all simple measurement paths, which can be solved in $O(|\mathcal{V}|)$ time [18].

- **Step 2 - Adding monitors to $\mathcal{M}$**: In the second step, MNMP checks if the identifiability requirements are being satisfied (i.e., $\Omega(\mathcal{N}) \geq k$) and expands the monitor set $\mathcal{M}$ in a greedy manner, with new monitors $w \in \mathcal{N}$ being iteratively added to $\mathcal{M}$ if they maximise the total sum of all node identifiability indexes $\Omega(n)$. This process leads to $O(|\mathcal{N}|^2)$ computations of $\Omega(n|\mathcal{M} \cup \{w\})$, i.e., the node identifiability indexes after selecting a new monitor, using the Ford-Fulkerson algorithm (explained in Section 3.6). This results in a time complexity of $O(|\mathcal{E}|^2|\mathcal{N}|^3)$ and $O(|\mathcal{M}||\mathcal{E}|^2|\mathcal{N}|^3)$ for CAP and CSP, respectively.
- **Step 3 - Remove Redundant Monitors**: The third and final step minimizes the monitor set obtained in step 2. MNMP iterates through all monitors $m \in \mathcal{M}$ and removes those whose deletion does not impact the maximum network identifiability index $\Omega(\mathcal{N})$. This requires $|\mathcal{M}||\mathcal{N}|$ computations of $\Omega(n|\mathcal{M} \setminus \{m\})$, i.e., the new node-identifiability index obtained by removing m, using Ford-Fulkerson. The overall complexity of this process is, consequently, $O(|\mathcal{M}||\mathcal{E}|^2|\mathcal{N}|^2)$ and $O(|\mathcal{M}|^2|\mathcal{E}|^2|\mathcal{N}|^2)$ for CAP and CSP, respectively.

Considering the above, it is possible to observe that, although providing a sub-optimal solution to the NP-hard monitor placement problem within polynomial time, MNMP remains relatively slow and certainly not suitable for practical usage, especially in large-scale systems. In addition, it is clear from the step-by-step time analysis that Step 2 is the most time consuming step of the algorithm. The high degree complexity time is due to the greedy search for the best monitor set, which results in a high number of computations of $\Omega(n)$.

Our initial experiments of MNMP show that, by executing the algorithm with our optimised implementation over a high-performance computer (details in Section 7), it still requires 2,268 seconds in average to find the minimum monitor set capable of identifying only two faults ($k = 2$) at the same time for a network with 604 nodes. Table 1 describes some results obtained for other networks of the same dataset [63] (details in Section 7).

Table 1. Computational time of MNMP for identifying two faults ($k = 2$) for networks with different sizes [63] and maximum 5% monitors from 10 independent executions.

Network	No. Nodes	No. Edges	Average Comp. Time (s)	Std. (s)
EXODUS	201	434	97.64	2.15
TELSTRA	318	758	425.09	1.98
SPRINTLINK	604	2268	2004.16	73.31

4 PROBLEM STATEMENT AND SOLUTION OVERVIEW

MNMP is a heuristic capable of finding the minimum monitor placement for any topology. However, due to its high execution time overhead, it cannot be applied to large-scale systems with changeable topologies, such as WSN. These changes in topology result in loss of connectivity and, consequently, changes in the auxiliary graphs, impacting the value of $\Omega(\mathcal{N})$. If $\Omega(\mathcal{N}) < k$, the current monitor placement $\mathcal{M}$ is not capable of uniquely identifying k node failures anymore, and new monitors need to be added to $\mathcal{M}$ to maintain the network's identifiability requirements.

Considering the above, for efficiently applying network tomography for node failure localisation in WSNs, we need to develop methods for (i) quickly identifying whether $\Omega(\mathcal{N}) \geq k$ given a topology change, and (ii) select new monitors to recover $\Omega(\mathcal{N})$ when necessary. In order to achieve

this, our first step is to reduce the computational complexity in solving the maximum flow problem. As explained in Section 3.6, the maximum flow has a time complexity of $O(|\mathcal{V}||\mathcal{E}|^2)$, and one way of improving the algorithm's performance is to reduce the number of edges in $\mathcal{G}$. This problem is defined as the *graph compression problem* below:

Problem 1 (Graph Compression $\mathcal{G} \rightarrow \hat{\mathcal{G}}$). *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, find a compressed graph $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ which is equivalent (in terms of monitor placement) to $\mathcal{G}$, where $|\hat{\mathcal{E}}| \leq |\mathcal{E}|$.*

The second step for efficiently finding $\mathcal{M}$ is to reduce the number of necessary operations when searching for new monitors. MNMP's heuristic approach considers every non-monitor $w \in \mathcal{N}$ as a candidate monitors, and compute the node-identifiability index $\Omega(n)$ for every other non-monitor $n \in \mathcal{N} \setminus \{w\}$, yielding $O(\mathcal{N}^2)$ computational complexity. Given a topology change, however, it is more efficient to update $\Omega(n)$ of only a subset of non-monitors affected by the change (i.e., $\Omega(n)$ < k), as well as only select candidate monitors capable of increasing the values of $\Omega(n)$ for these affected non-monitors, instead of iterating through all non-monitors. Given these observations, we define the following problems:

Problem 2 (Set of Affected Non-Monitors $\mathcal{A}$). *Given $\mathcal{G}$ and a monitor set $\mathcal{M}$, find a set of non-monitors $\mathcal{A}$ where the $\Omega(n)$ of every $n \in \mathcal{A}$ is smaller than a given identifiability requirement k.*

Problem 3 (Smallest Set of Monitor Candidates C). *Given a set of affected non-monitors $\mathcal{A}$, find the smallest set of monitor candidates $C \subseteq \mathcal{N}$ from which we are able to select enough monitors that recovers $\Omega(n)$ of every $n \in \mathcal{A}$ to k.*

By solving these three problems, we can reduce the computational complexity of MNMP from $O(|\mathcal{E}||\mathcal{N}|^3)$ to $O(|\hat{\mathcal{E}}|^2|\mathcal{N}||\mathcal{A}||C|)$, where $|\hat{\mathcal{E}}| \ll |\mathcal{E}|$, $|\mathcal{A}| \ll |\mathcal{N}|$ and $|C| \ll |\mathcal{N}|$. Further details will be given in the following sections.

Table 2 summarises the notations used throughout this paper.

5 GRAPH COMPRESSION WITH DUMMY NODES

Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, MNMP finds the best monitor placement $\mathcal{M}$ by computing the maximum network identifiability index $\Omega(\mathcal{N})$ in the auxiliary graph $\mathcal{G}^*$. This process requires the computation of the node identifiability indexes $\Omega(n)$ of all non-monitors using the Ford-Fulkerson algorithm for the maximum flow problem. As discussed in Section 3.6, Ford-Fulkerson has the computational complexity of $O(|\mathcal{V}||\mathcal{E}|^2)$, and one way of improving its performance is to reduce the number of edges by compressing the graph. It is worth noticing, however, that this graph compression must maintain the value of the maximum flow of all non-monitors, so the value of $\Omega(\mathcal{N})$ and $\mathcal{M}$ obtained for the compressed graph $\hat{\mathcal{G}}$ is the same as the original graph $\mathcal{G}$.

Considering this, we use *Edge Concentration* in our graph compression solution. Edge concentration (EC) is an algorithm capable of identifying correlations between the edges of neighbour nodes, aggregating those edges sharing similar neighbourhood [57]. It has been recently used by network mining for simplifying large-scale networks analysis [12, 54]. Network mining aims to extract recurring patterns in large and complex network-based systems, such as Biological databases, and Edge Concentration is efficiently employed to compress these graphs due to the semantic and/or spatial/temporal correlation between nodes.

EC aggregates edges by first finding the bicliques contained in a bipartite graph. This process is necessary for easily identifying the similar edges and inserting special nodes called *dummy nodes* that replace the aggregated edges. For a better understanding of this method, we briefly introduce the concepts of *induced bipartite graph (bigraph)* and *bicliques* in the following sections.

Table 2. List of notations

SYMBOL	DESCRIPTION
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	Set of network/nodes/links, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$\mathcal{M}, \mathcal{N}$	Set of monitors / non-monitors, where $\mathcal{M} \cup \mathcal{N} = \mathcal{V}$, and $\mathcal{M} \cap \mathcal{N} = \emptyset$ (Section 3.1)
k	Systems' identifiability requirement defining the maximum number of simultaneous failures.
$\Omega(\mathcal{N})$, $\Omega(n)$	Maximum network identifiability index and node identifiability index for a non-monitor $n \in \mathcal{N}$, where $\Omega(\mathcal{N}) = min_{n \in \mathcal{N}} \Omega(n)$ (Section 3.4, Definition 3).
m'	Virtual monitor representing all the monitors in the original network (i.e. $\mathcal{M} \subseteq \mathcal{G}$) (Section 3.5)
$\mathcal{G}^*$	Auxiliary graph (Eq.(1)) obtained from $\mathcal{G}$ given m' and under CAP (Sections 3.3 and 3.5)
$\mathcal{G}^*_m$	Auxiliary graph (Eq.(2)) obtained from $\mathcal{G}$ given a source monitor $m \in \mathcal{M}$, $m' = \mathcal{M} \setminus m$, and under CSP (Sections 3.3 and 3.5)
$\hat{\mathcal{G}}$	A graph compressed from $\mathcal{G}$ via Edge Concentration (Section 5).
$\mathcal{A}_0$	Set of non-monitors whose identifiability index $\Omega(n), \forall n \in \mathcal{A}_0$ *may* be affected by a topology change (Section 6)
$\mathcal{A}$	Set of non-monitors affected by a topology change ($\Omega(n) < k, \forall n \in \mathcal{A}$) (Section 6)
C	Set of candidate monitors which can increase $\Omega(\mathcal{A})$ (Section 6.2)

5.1 Induced Bigraph

The first step of EC is to construct *an induced bipartite graph (bigraph)* from $\mathcal{G}$, which is defined as follows:

Definition 4. *An induced bipartite graph (bigraph) from a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a bipartite graph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$, such that its two disjoint node sets $\mathcal{T} = \{x \in \mathcal{V} \mid \mathcal{O}(x) \neq \varnothing\}$, $\mathcal{B} = \{x \in \mathcal{V} \mid \mathcal{I}(x) \neq \varnothing\}$,* [1] *and for each $u \in \mathcal{T}$ and $v \in \mathcal{B}$, $(u, v) \in \tilde{\mathcal{E}}$ if and only if there is an edge from u to v in $\mathcal{G}$.*

Intuitively, an induced bigraph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$ visualizes the neighborhood structure of $\mathcal{G}$ differently, dividing the neighborhood into nodes with out-edges (in-neighbours) and nodes with in-edges (out-neighbours) to better trace different paths arriving and leaving a defined node x.

Note that, when node x has both in- and out-neighbors in $\mathcal{G}$, the label x that appears in both $\mathcal{T}$ and $\mathcal{B}$ is regarded as two distinct nodes despite the same label. To avoid ambiguity, we shall use $x \in \mathcal{T}$ and $x \in \mathcal{B}$ to distinguish them. Each directed edge in $\mathcal{G}$ is mapped to one edge in $\tilde{\mathcal{G}}$, and thus, $|\mathcal{E}| = |\tilde{\mathcal{E}}|$. For instance, the left part of Figure 3 shows the induced bigraph $\tilde{\mathcal{G}}$ from $\mathcal{G}$. From

[1] The notation $\mathcal{O}(x)$ denotes the out-neighbours set of node x, while $\mathcal{I}(x)$ the in-neighbours set of the same node.

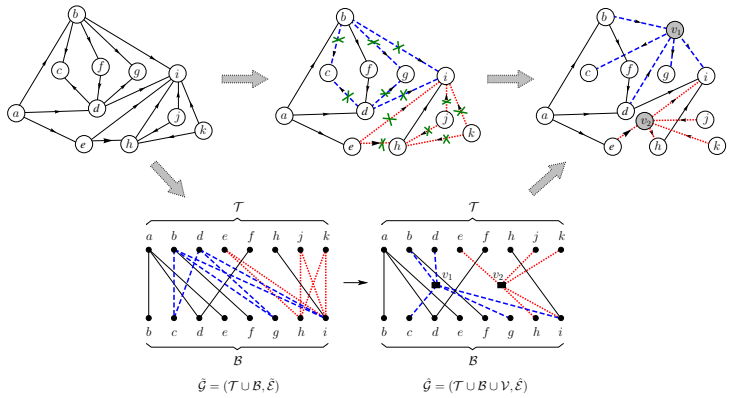


Fig. 3. Compression of Induced Bigraph $\tilde{\mathcal{G}}$ into $\hat{\mathcal{G}}$ via Edge Concentration EC. The blue edges share similar source nodes (b and d) and target nodes (c, g, and i) and, therefore, belong to the same biclique. These edges can be aggregated by the dummy node v_1. Similarly, edges in red belongs to another biclique and are aggregated by the dummy node v_2

$\tilde{\mathcal{G}}$, we can clearly see that b and d in $\mathcal{B}$ are both connected with a in $\mathcal{T}$, meaning that, in $\mathcal{G}$, b and d both have an in-neighbor a.

5.2 Bicliques Identification.

Based on the induced bigraph $\tilde{\mathcal{G}}$, we next introduce the notion of *bipartite cliques (bicliques).*

Definition 5. *Given an induced bigraph $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$, a pair of two disjoint subsets $\mathcal{X} \subseteq \mathcal{T}$ and $\mathcal{Y} \subseteq \mathcal{B}$ is called a biclique if $(x, y) \in \tilde{\mathcal{E}}$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.*

Intuitively, a biclique $(\mathcal{X}, \mathcal{Y})$ is a complete bipartite subgraph of $\tilde{\mathcal{G}}$, which has $|\mathcal{X}| + |\mathcal{Y}|$ nodes and $|\mathcal{X}| \times |\mathcal{Y}|$ edges. Each biclique $(\mathcal{X}, \mathcal{Y})$ in $\tilde{\mathcal{G}}$ tells us that in $\mathcal{G}$, all nodes $y \in \mathcal{Y}$ have the common in-neighbor set $\mathcal{X}$. For example, there are two bicliques in Figure 3: $(\{b, d\}, \{c, g, i\})$ in dashed line, and $(\{e, j, k\}, \{h, i\})$ in dotted line. Biclique $(\{b, d\}, \{c, g, i\})$ in $\tilde{\mathcal{G}}$ implies that in $\mathcal{G}$, three nodes c, g, i all have two in-neighbors $\{b, d\}$ in common.

Bicliques are necessary to compress a bigraph $\tilde{\mathcal{G}}$ into a *dummy graph* $\hat{\mathcal{G}}$ since the main ingredient of edge concentration is to group sets of edges in $\tilde{\mathcal{G}}$ together to reduce the total number of edges while retaining the same information as the original graph. Unfortunately, it has been proved that EC is a NP-hard problem [35] and in order to compress $\tilde{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B}, \tilde{\mathcal{E}})$ in a efficient way we leverage Buehrer and Chellapilla's algorithm [16], which is capable of finding collections of bicliques in $\tilde{\mathcal{G}}$ with a computation complexity of only $O(|\tilde{\mathcal{E}}| \log(|\mathcal{T}| + |\mathcal{B}|))$.

The central idea underpinning Buehrer and Chellapilla's algorithm [16] is using virtual-node compression to provide a succinct representation of the graph constructed by replacing dense subgraphs by sparse ones. A frequent-pattern mining algorithm was introduced for finding directed bipartite cliques. A directed bipartite clique on the vertex set K is replaced by a star centred at a new "virtual" node, with nodes in K being the leaves. Applying this transformation repeatedly leads to a compressed graph with significantly fewer edges and a relatively small number of additional nodes.

Buehrer and Chellapilla's algorithm [16] has been demonstrated as highly effective in that the virtual-node compression can achieve high compression ratios on a variety of graphs. This high compression ratio reflects the frequent occurrence of bipartite cliques in real-world graphs (e.g.,

social networks, web graphs, infrastructure networks), whose results were consistent with those in the context of community-finding over social graphs.

We then replace edges of each biclique $(\mathcal{X}, \mathcal{Y})$ with a special node, called *dummy node*, whose "fan-in"(i.e., the incoming nodes) is all nodes in $\mathcal{X}$ and whose "fan-out" (i.e., the outgoing nodes) is all nodes in $\mathcal{Y}$. Finally, the compressed graph, denoted as $\hat{\mathcal{G}} = (\mathcal{T} \cup \mathcal{B} \cup \hat{\mathcal{V}}, \hat{\mathcal{E}})$, can be obtained from bigraph $\tilde{\mathcal{G}}$, where $\mathcal{T}$ and $\mathcal{B}$ are the same as those of $\tilde{\mathcal{G}}$, $\hat{\mathcal{V}}$ is the set of edge concentration nodes, and $\hat{\mathcal{E}}$ is the set of edges in $\hat{\mathcal{G}}$.

In practice, $|\hat{\mathcal{E}}|$ is typically much smaller than $|\tilde{\mathcal{E}}|$, since $|\mathcal{X}| \times |\mathcal{Y}|$ edges of each biclique in $\tilde{\mathcal{G}}$ are reduced to $|\mathcal{X}| + |\mathcal{Y}|$ edges in $\hat{\mathcal{G}}$, which is a substantial improvement achieved by edge concentration. The right part of Figure 3, for example, depicts the resultant graph $\hat{\mathcal{G}}$ of applying this approach to $\tilde{\mathcal{G}}$.

It is worth mentioning, that the dummy nodes **are not considered** in the computation of the network identifiability index $\Omega(\mathcal{N})$ since they are only *auxiliary elements* employed by the heuristic proposed in [16]. In terms of monitor placement, the final graph obtained, $\hat{\mathcal{G}}$, is a more concise structure, with the same number of non-monitors and less edges. With graph compression, the computation of the monitor placement has, therefore, a time complexity of $O(|\hat{\mathcal{E}}|^2||\mathcal{N}|^3)$.

5.3 Edge Weight Assignment

Although the compressed graph $\hat{\mathcal{G}}$ can effectively reduce the number of edges in the original graph $\mathcal{G}$ through edge concentration, it is not known whether the maximum flow of the original and compressed graphs are well preserved. In this section, we will show that, without correctly assigning weights to some edges, the maximum flow of the compressed $\hat{\mathcal{G}}$ always lower-bounds the maximum flow of the original one. More importantly, when the appropriate edge weights are assigned to $\hat{\mathcal{G}}$, the maximum flow of the weighted compressed graph $\hat{\mathcal{G}}$ is equal to the maximum flow of $\mathcal{G}$.

Our key observation is that, after edge concentration, the number of edges in each biclique $(\mathcal{X}, \mathcal{Y})$ of the the original graph $\mathcal{G}$ reduces from $|\mathcal{X}| \times |\mathcal{Y}|$ to $|\mathcal{X}| + |\mathcal{Y}|$ in $\hat{\mathcal{G}}$. Thus, the sum of the output flow over all nodes of $\mathcal{X}$ to $\mathcal{Y}$ in $\mathcal{G}$ is equal to the flow into the dummy node z of the biclique $(\mathcal{X}, \mathcal{Y})$ in $\hat{\mathcal{G}}$. Similarly, the sum of the input flow over all nodes of $\mathcal{Y}$ from $\mathcal{X}$ in $\mathcal{G}$ is equal to the flow out of the dummy node z of the biclique $(\mathcal{X}, \mathcal{Y})$ in $\hat{\mathcal{G}}$. To guarantee the maximum flow of the compressed graph $\hat{\mathcal{G}}$ to be exactly the same as the maximum flow of the original graph, we need to increase the capacity of the edges in each biclique of $\hat{\mathcal{G}}$, as opposed to the $\mathcal{G}$, whose edge weights are all equal 1.

We propose, therefore, an *edge weight assignment* method for the compressed $\hat{\mathcal{G}}$ based on the following principles:

(1) We first scan each dummy node z in $\hat{\mathcal{G}}$, and calculate the number of "fan-in" nodes of z, denoted as $deg(z)^-$, and the number of "fan-out" nodes of z, denoted as $deg(z)^+$.
(2) Then, for every dummy node z in $\hat{\mathcal{G}}$, let

$$\mathcal{I}(z) = \{(x, z) | x \in \text{all the "fan-in" nodes of } z\}$$
$$O(z) = \{(z, x) | x \in \text{all the "fan-out" nodes of } z\}$$

we assign each edge $e^- \in \mathcal{I}(z)$ a weight $w(e^-) = deg(z)^+$, and each edge $e^+ \in O(z)$ a weight $w(e^+) = deg(z)^-$.
(3) Finally, all the edges in $\hat{\mathcal{G}}$ whose either end point is not a dummy node are assigned weight equal 1.

Following the above rules for edge weights assignment in the compressed $\hat{\mathcal{G}}$, one can readily show that the maximum flows of the compressed $\hat{\mathcal{G}}$ and the original $\mathcal{G}$ are the same, based on the following theorem:

Theorem 1. *Given a network $\mathcal{G}$, the network identifiability $\Omega(\mathcal{G}) = \Omega(\hat{\mathcal{G}})$, where $\Omega(\hat{\mathcal{G}})$ is the weighted dummy graph compressed from $\mathcal{G}$.*

Proof. Let (X_i, Y_i) be the i-th biclique of the original graph $\mathcal{G}$, and (X_i, Y_i, z_i) be the i-th compressed star subgraph of the biclique (X_i, Y_i) in the compressed graph $\hat{\mathcal{G}}$. Since the only difference of the edge capacity between the original and compressed graphs is in each biclique replacement, we will show that the maximum flow of each (X_i, Y_i) in $\mathcal{G}$ is the same as the maximum flow of (X_i, Y_i, z_i) in $\hat{\mathcal{G}}$.

For each biclique (X_i, Y_i) in the original graph $\mathcal{G}$, since the capacity of each edge is 1, the sum of the capacity from each node x in X_i to all nodes of Y_i is a constant, $|Y_i|$. According to our edge weight assignment rule, after edge concentration, to guarantee the same flow from each x in X_i to the dummy node z_i in the compressed $\hat{\mathcal{G}}$, the capacity of the edge e^- from each node x in X_i to z_i is $\deg(z_i)^+$. Thus, the sum of the maximum flow from all nodes in X_i to dummy node z_i is

$$\begin{aligned}\sum_{e^- \in \{(x,z_i)\}_{x\in X_i}} \text{capacity}(e^-) &= \sum_{e^- \in \{(x,z_i)\}_{x\in X_i}} \deg(z_i)^+ = \deg(z_i)^+ \left(\sum_{e^- \in \{(x,z_i)\}_{x\in X_i}} 1 \right) \\ &= \deg(z_i)^+ |X_i| = \sum_{x\in O(z_i)} |X_i| = |X_i||Y_i|\end{aligned} \tag{4}$$

Similarly, one can readily show that the sum of the maximum flow from dummy node z_i to all nodes in Y_i satisfies

$$\sum_{e^+ \in \{(z_i,y)\}_{y\in Y_i}} \text{capacity}(e^+) = \sum_{x\in I(z_i)} |Y_i| = |X_i||Y_i|$$

Thus, it follows that

$$\sum_{e^- \in \{(x,z_i)\}_{x\in X_i}} \text{capacity}(e^-) = |X_i||Y_i| = \sum_{e^+ \in \{(z_i,y)\}_{y\in Y_i}} \text{capacity}(e^+)$$

Since $|X_i||Y_i|$ is the total number of edges in each biclique of the original graph $\mathcal{G}$, the maximum flow of $\mathcal{G}$ is equal to the maximum flow of the weighted compressed $\hat{\mathcal{G}}$. □

6 ADAPTIVE MONITOR PLACEMENT

By transforming $\mathcal{G}$ into a dummy graph $\hat{\mathcal{G}}$, we have successfully reduced the overhead of the maximum-flow problem. The next problem is: when topology changes occur, is there a way of quickly (i) identifying if the network's identifiability requirements are still valid (i.e., $\Omega(\mathcal{N}) \geq k$), and (ii) selecting new monitors to recover the value of $\Omega(\mathcal{N})$ when necessary. Algorithm 1 demonstrates the operation of our adaptive monitor placement (AMP) algorithm. AMP first identifies an initial affected set $\mathcal{A}_0 \subseteq \mathcal{N}$, where the identifiability index $\Omega(n \in \mathcal{A}_0)$ for may be affected by topology changes. $\mathcal{A}_0$ is then further reduced to a final affected set $\mathcal{A} \subseteq \mathcal{A}_0$, where $\Omega(n) < k$ for every $n \in \mathcal{A}$. Based on $\mathcal{A}$, AMP identifies candidate monitors set C, which is necessary for efficiently recovering the maximum network identifiability index $\Omega(\mathcal{N})$ to its desired level k. Exploiting these monitors C, AMP recovers network identifiabiliy index to its desired level $\Omega(\mathcal{N}) \geq k$. The details of finding $\mathcal{A}$ and C are described in Subsections 6.1 and 6.2, respectively.

6.1 Identify the Set of Affected Non-Monitors

Before describing our solutions, it is important to define a topology change:

Definition 6 (Topology change). *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a topology change consists of a set of edges $\mathcal{E}_R$ or a set of nodes $\mathcal{V}_R$ that are deleted from the network $\mathcal{G}$ due to link interference or nodes shutting down.*

WSNs are more susceptible to topology changes compared to other networked systems given the constant system-environment interaction (since the devices are placed near the phenomenon of interest) and the devices' resource constraints (e.g. limited power supply) [7, 37, 68]. These changes result in loss of connectivity in parts of the network and are usually caused by link interference (e.g., malicious attacks like jamming or weather condition variations) or node's energy depletion.

As explained in Section 3, $\Omega(\mathcal{N})$ is obtained by computing the node identifiability index $\Omega(n)$ of all non-monitors in the auxiliary graph $\mathcal{G}^*$. Topology changes, however, lead to variations in $\mathcal{G}^*$ and, consequently, the value of $\Omega(n)$. Considering that $\Omega(\mathcal{N}) = \min \Omega(n),\ \forall n \in \mathcal{N}$ (Definition 3), the most intuitive way of identifying if the value of $\Omega(\mathcal{N})$ changed is, therefore, to recompute $\Omega(n)$ for every non-monitor $n \in \mathcal{N}$. If any of these $\Omega(n)$ is lower than the identifiability requirement defined by k, new monitors need to be added to the initial monitor set.

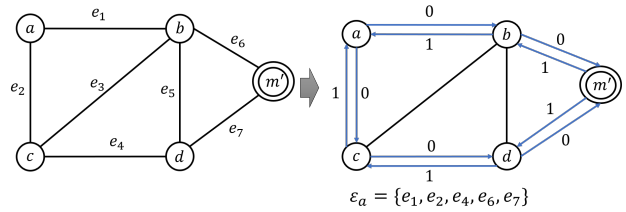
This iterative operation is, however, very inefficient as there exist many redundant computations. For recovery purposes, it is sufficient to examine *only* the non-monitors *affected* by the topology change(i.e., the edges in $\mathcal{E}_R$ or the nodes in $\mathcal{V}_R$). To better understand our proposed solution, the affected set identification problem is formulated as follow:

Definition 7 (Affected non-monitor set $\mathcal{A}$). *Given a network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the current monitor set $\mathcal{M}$ and non-monitor set $\mathcal{N} = \{\mathcal{V} \setminus \mathcal{M}\}$, and a set of deleted edges $\mathcal{E}_R$, $\mathcal{A} \subseteq \mathcal{N}$ consists of the set of non-monitors which identifiability index is below the required identifiability value k, $\Omega(n) < k,\ \forall n \in \mathcal{A}$.*

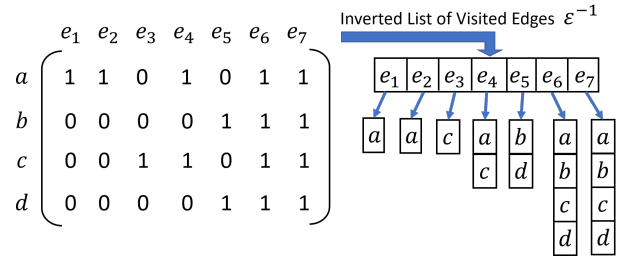
Remark 1. *It is worth noting that topology changes caused by a set of deleted nodes $\mathcal{V}_R$ can be modelled as multiple deleted edges sets $\mathcal{E}_{R_{v_1}}, \mathcal{E}_{R_{v_2}}, ..., \mathcal{E}_{R_{v_k}}, \{v_1, v_2, .., v_k\} \in \mathcal{V}_R$, and the same approach provided can be applied seamlessly.*

In order to find $\mathcal{A}$ without recomputing $\Omega(n)$ for every $n \in \mathcal{N}$, we first record the paths that each non-monitors n uses when computing $\Omega(n)$ in the initial monitor placement. This process can be done by analysing the residual graphs obtained from the computation of the maximum flow of each non-monitors. A residual graph $\tilde{\mathcal{G}}_n^*$ is a graph containing the non-saturated paths with the final available capacity (i.e., the difference between the initial edge capacity and the flow passing through it). For each non-monitor node n, a new residual graph is generated describing the paths used by n after the computation of it's maximum-flow. To reduce recovery time and storage space, these paths can be recorded in a flow matrix $\mathbf{L}$ as illustrated in Figure 4, where element $\mathbf{L}_{i,j}$ is equal 1 if edge j is traversed by a flow starting from node i and 0 otherwise. With $\mathbf{L}$, we can quickly identify a set of *initial affected non-monitors* $\mathcal{A}_0$ whose flow traverse the deleted edge $e \in \mathcal{E}_R$ by corresponding elements $\mathbf{L}_{e,i}, \forall i \in \mathcal{N}$.

Since $\mathbf{L}$ is typically a sparse table (i.e. most of the elements therein are 0), we can further reduce space and processing time by turning this table into an *inverted index of visited edges*, $\mathbf{L}^{-1}$, a data structure widely used by database community for fast information retrieval [8]. $\mathbf{L}^{-1}$ can be regarded as a hashed map where each entry $\mathbf{L}_i^{-1}$ consists of a list of non-monitors that will be affected by the



(a) An auxiliary graph $\mathcal{G}^*$ and one example of residual graph $\tilde{\mathcal{G}}^*$



(b) Flow matrix correlating visited edges and nodes and its corresponding inverted list $\mathcal{E}^{-1}$

Fig. 4. Example of auxiliary graph $\mathcal{G}^*$ for a network with 4 non-monitors and 7 edges (indexes $e_1, e_2, e_3, e_4, e_5, e_6, e_7$). Figure 4(a) shows the residual graph $\mathcal{G}_R^*$ and the list of visited edges $\mathcal{E}_a$ obtained by computing the maximum flow of node a. Figure 4(b) shows the flow-matrix and inverted list of visited edges generated by repeating the same process for the other nodes.

potential removal of edge e (which is the label of that entry). As can be seen, this structure further reduces storage and computational complexity to $O(|\mathcal{E}||\mathcal{A}_0|)$ and $O(|\Delta\mathcal{E}||\mathcal{A}_0|)$, respectively, where the number of edge removed is denoted by $\Delta\mathcal{E}$. Consequently, by simply recomputing the $\Omega(n)$ for those $n \in \mathcal{A}_0 \subseteq \mathcal{N}$, it is possible to obtain the *final affected non-monitors* $\mathcal{A} \subseteq \mathcal{A}_0$ where $\Omega(n) < k, \forall n \in \mathcal{A}$.

Algorithm 1 Adaptive Monitor Placement

Input: inverted List $\mathbf{L}^{-1}$, residual graphs $\mathbb{G} = \{\tilde{\mathcal{G}}_1^*, \cdots, \tilde{\mathcal{G}}_{|\mathcal{A}|}^*\}$, removed edge e, network identifiability reqirement k, current monitor set $\mathcal{M}$
Output: updated monitor set $\mathcal{M}'$

$\mathcal{M}' \leftarrow \emptyset, \mathcal{A} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset$
$\mathcal{A} \leftarrow \mathbf{L}_e^{-1}$; ▷ Initial set of affected nodes
for $n \in \mathcal{A}$ **do**
  if $\Omega(n) \geq k$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \setminus n$;
  end if
end for
if $\mathcal{A} \neq \emptyset$ **then** ▷ $\Omega(n) < k,\ \forall n \in \mathcal{A}$
  $\mathcal{C} \leftarrow$ FIND-C($\mathbb{G}, \mathcal{A}, e, k$)
  $\mathcal{M}' \leftarrow$ run MNMP Step 2 with $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{M}$
end if
return $\mathcal{M}'$

6.2 Reduction of Monitor Candidates

By exploiting graph compression to reduce the network (Section 5) and identifying affected non-monitors set $\mathcal{A}$ (Subsection 6.1) given a topology change, we have successfully reduced the computational complexity of the monitor placement problem to $O(|\hat{\mathcal{E}}|^2|\mathcal{N}|^2|\mathcal{A}|)$. The remaining unsolved question is: *can we further reduce the number of possible candidates to join the monitor set and, consequently, the monitor placement execution time?*

In the traditional solution, all non-monitors are regarded as potential candidates for the monitor set $\mathcal{M}$ and, therefore, tested in every iteration. In adaptive scenarios, however, we observe that many of these candidates are redundant and unnecessary since many of them have no actual contribution to the nodes in $\mathcal{A}$. To this end, our aim here is to find a smallest set of monitor candidates C which can fulfil the following condition:

$$\begin{aligned} &\text{minimize} && |C| \\ &\text{subject to} && \exists\, c \in C : p_{n,c} \text{ exists},\ n \in \mathcal{A} \end{aligned} \tag{5}$$

As can be seen, while minimizing the number of monitor candidates in C, it ensure that every candidate therein contributes to increasing the $\Omega(n)$ of at least one affected non-monitor $n \in \mathcal{A}$.

Given $\mathcal{A}$ and a collection of the residual graphs $\tilde{\mathcal{G}}_n^*$ where $n \in \mathcal{A}$, we first use Breadth-First Search (BFS) to find a set non-monitors C_n, where a path $p_{n,c}$ exists from n to c for all $c \in C_n$, for each $n \in \mathcal{A}$. With $|\mathcal{A}|$ number of C_n, we define C as below:

$$C \coloneqq \bigcup_{n \in \mathcal{A}'} C_n \tag{6}$$

C consists of all non-monitors where at least one path exists from $\mathcal{A}$. Based on C, a collection of set $\mathbb{S} = \{\mathcal{S}_1, \cdots, \mathcal{S}_{|C|}\}$ can be generated. Each set $\mathcal{S}_c \in \mathbb{S}$ consists of affected non-monitors $n \in \mathcal{A}$ from which exists an augmenting path $p_{n,c}$ to c. Here we further refine $\mathbb{S}$ by removing redundant sets. When $\mathcal{S}_{c1} \subseteq \mathcal{S}_{c2}$, $\mathcal{S}_{c2}$ is removed from $\mathbb{S}$, for example. If $\mathcal{S}_{c1} = \mathcal{S}_{c2}$, $c2$ is also recorded alongside $c1$. After this refining process, every set in $\mathbb{S}$ becomes a distinctive set, that is, none of the sets in $\mathbb{S}$ is a subset of another.

Figure 5 illustrates an example of this operation. Given a $\mathcal{A} = \{n_1, n_2, n_3, n_4\}$ and their corresponding sets of candidates $C_{n1} = \{c_1, c_2\}$, $C_{n2} = \{c_2, c_3, c_4, c_5\}$, $C_{n3} = \{c_1, c_2, c_3, c_5\}$, $C_{n4} = \{c_3, c_4, c_5\}$,, a collection $\{\mathcal{S}_{c1}, \cdots, \mathcal{S}_{c5}\}$ is generated. In this collection, $\mathcal{S}_{c1} \subset \mathcal{S}_{c2}$ and $\mathcal{S}_{c4} \subset \mathcal{S}_{c3}$ are redundant sets that can be ignored while $\mathcal{S}_{c5} = \mathcal{S}_{c3}$ is also removed but with its label c_5 recorded. Consequently, we have a collection of set $\mathbb{S} = \{\{n_1, n_2, n_3\}, \{n_2, n_3, n_4\}\}$, of which the universe of this collection is $\mathcal{A} = \{n_1, n_2, n_3, n_4\}$.

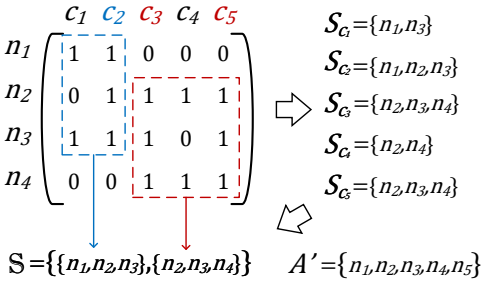


Fig. 5. Convert monitor candidate selection problem to a set cover problem (SCP) with $\mathbb{S}$ and $\mathcal{A}$

With $\mathbb{S}$, the problem stated in Eq. (5) can now be reduced to another problem: selecting the smallest sub-collection of $\mathbb{S}$ whose union consists of all elements in $\mathcal{A}$. This is known as the classic Set Cover Problem (SCP) defined below:

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathcal{S}\in\mathbb{S}} x_{\mathcal{S}} \\ \text{subject to} \quad & \sum_{\mathcal{S}:n\in\mathcal{S}} x_{\mathcal{S}} \geq 1,\ \forall n \in \mathcal{A} \\ & x_{\mathcal{S}} \in 0,1\ \forall \mathcal{S} \in \mathbb{S} \end{aligned} \tag{7}$$

SCP is a well studied problem proven to be NP-complete. Fortunately, there exists an effective greedy solution where at each iteration, it chooses the set in $\mathbb{S}$ containing the largest number of uncovered elements in $\mathcal{A}$, detail of which is described in Alg. 2. With the collection selected from $\mathbb{S}$, their corresponding nodes $c \in \bar{C}$ are select as monitor candidates. Fig. 5 demonstrates an example where $C = \{c_2, c_3, c_5\}$ is the set of monitor candidates. The computational complexity of this algorithm is given by $O(|\hat{\mathcal{E}}||\mathcal{N}||\mathcal{A}|)$. It is worth noting that C can be used under both CAP and CSP since $\mathcal{G}^*_m \subseteq \mathcal{G}^*$, where the connectivity in between affected node and monitor candidates remains unchanged.

Algorithm 2 Find Candidate Monitors FIND-C($\mathbb{G}, \mathcal{A}, e, k$)

Input: residual graphs $\mathbb{G} = \{\tilde{\mathcal{G}}^*_1, \cdots, \tilde{\mathcal{G}}^*_{|\mathcal{A}|}\}$, affected non-monitors $\mathcal{A}$, removed edge e, network identifiability reqirement k
Output: candidate monitor set C

$C \leftarrow \varnothing, \mathbb{S} \leftarrow \varnothing, \mathbb{T} \leftarrow \varnothing$
for each $n \in \mathcal{A}$ **do**
  $C_n \leftarrow \mathsf{BFS}(\tilde{\mathcal{G}}^*_n, n)$ ▷ candidates increases $\Omega(n)$
end for
$C \leftarrow \bigcup_{n\in\mathcal{A}} C_n$
for each $c \in C$ **do**
  $\mathcal{S}_c \leftarrow \{n : c \in C_n, \forall n \in \mathcal{A}\}$
  $\mathbb{S} \leftarrow \mathbb{S} + \mathcal{S}_c$
  $\mathcal{T}_c \leftarrow \varnothing$
  $\mathbb{T} \leftarrow \mathbb{T} + \mathcal{T}_c$
end for
/*Remove redundant sets from $\mathbb{S}$*/
for each $\mathcal{S}_c \in \mathbb{S}$ **do**
  if $\mathcal{S}_c \subseteq \mathcal{S}_x,\ \forall \mathcal{S}_x \in \mathbb{S}$ **then**
    $C \leftarrow C \setminus c\ , \mathbb{S} \leftarrow \mathbb{S} \setminus \mathcal{S}_c\ , \mathbb{T} \leftarrow \mathbb{T} \setminus \mathcal{T}_c$
    if $\mathcal{S}_c = \mathcal{S}_x$ **then**
      $\mathcal{T}_x \leftarrow \mathcal{T}_x \cup c$
    end if
  end if
end for
$C \leftarrow$ Solve SCP problem with $\mathbb{S}$ and $\mathcal{A}$ and return set labels from C and $\mathbb{T}$
return C

After finding C, AMP only need to search in C instead of $\mathcal{N}$ for new monitors. We can conclude that the computational complexity of AMP given by $O(|\hat{\mathcal{E}}|^2|\mathcal{N}||\mathcal{A}||C|)$ ($O(|\mathcal{M}||\hat{\mathcal{E}}|^2|\mathcal{N}||\mathcal{A}||C|)$ in

the case of CSP), which is upper bounding by MNMP's computational complexity $O(|\hat{\mathcal{E}}|^2|\mathcal{N}|^3$ since $\mathcal{A} \subseteq \mathcal{N}$ and $C \subseteq \mathcal{N}$. In the next Section (Evaluation), our experiments shows both that $|\mathcal{A}|, |C| \ll |\mathcal{N}|$ and that the monitor placement is obtained in few minutes with our solution, compared to hours using the traditional heuristic. As a trade off, the number of monitors selected by AMP is lower bound by MNMP since AMP only searches for candidates in a subset of non-monitors. However, according to our observation during experiments, the difference is small. To demonstrate this, we ran an 100-round experiment based on ABOVE and EBONE in Table 3 3. In each round, we randomly removed 10% edges from the network topology. The results show that AMP can recover network identifiability index by selecting the same numbers of new monitors.

7 EVALUATION

In this section, we present experiments analysing the performance of our proposed solutions, Graph Compression and Adaptive Monitor Placement (AMP), for different network topologies. All the algorithms are implemented in MATLAB R2019a and the experiments are performed on a high performance computer (HPC) provided by the university. This HPC consists of 64 cores mainly composed of dual Xeon E5-series processors, with 500GB of memory, local magnetic storage and 1Gb Ethernet interconnect. It is worth noting that although HPC was used for our experiment, MATLAB R2019a used only one core per thread. We use HPC just to run the experiments in parallel. The experiment results, consequently, are comparable to those on general personal desktops.

The remainder of this section is organised as follows. We first detail the datasets used in our experiments in Subsection 7.1 and validate our proposed Graph Compression algorithm in Subsection 7.2. In Subsection 7.3, we evaluate AMP by testing its ability to compute the final affected set $\mathcal{A}$ and candidate monitors C. The overall computational overhead of AMP against MNMP is then presented in Subsection 7.4.

7.1 Experimental Datasets

With the aim of analysing our solutions in a variety of scenarios, we utilise both real-world and synthetic network topologies in these experiments. Each topology is detailed below.

7.1.1 ***Real-World Data Sets.*** In terms of real-world topology, we exploit the Rocketfuel data set [63]. This dataset is widely used in many monitor placement papers and is well-known to the community, making it an ideal dataset for better *comparing* our solutions with the state-of-the-art heuristic, MNMP. It consists of 9 different networks, as detailed in Table 3, representing a great variety of topology settings, both in terms of scale and connectivity.

7.1.2 ***Synthetic Random Graphs.*** In terms of synthetic datasets, we generated different types of random graphs based on three models: **Erdõs-Rényi (ER)** [22], **Barabasi-Albert (BA)** [4], and **Watts-Strogatz (WS)** [58]. A brief description of each of these models is listed as follow:

- **Erdõs-Rényi (ER) Model**. Generates random networks with the node degree following a binomial distribution. Considering a network with isolated nodes, ER gradually builds the graph by randomly adding edges between pairs of nodes with a given probability p. This is the simplest of the random network models but very popular in many publications, and it is used here as a baseline for testing the proposed solution.
- **Barabasi-Albert (BA) Model**. Given a parameter M, this model can generate scale-free networks whose degree distribution follows a power law, which is widely observed in some practical networks, such as the Internet, Power Grids, and financial networks. These networks follow a preferential attachment model, where the majority of vertexes have only few numbers of connected edges, while only few vertexes have large numbers of connected edges.

Table 3. Real-World Topologies Data Set [63]

Network	No. Edges	No. Nodes	Avg. Degree	Diameter
ABOVE	294	182	3.238	12
EBONE	381	172	4.43	12
TISCALI	404	240	3.36	14
EXODUS	434	201	4.31	11
TELSTRA	758	318	4.76	13
ATT	2078	631	6.58	10
SPRINTLINK	2268	604	7.5	11
VERIO	2821	960	5.87	13
LEVEL3	5298	624	16.98	10

- **Watts-Strogatz (WS) Model**. Given a mean degree K and β, this model produces networks with small-world properties, where the average distance between vertexes is very small compared to the size of networks. This behaviour is common to many empirical graphs describing human relationships, such as social networks and telephone call graphs. Both the underlying architecture of the internet and the relationship between websites in the World Wilde Web can also be modeled through WS.

7.2 Experiment Results of Graph Compression $\mathcal{G} \Rightarrow \hat{\mathcal{G}}$

In this set of experiments, we exam our graph compression scheme from three different aspects: (i) the graph compression rate (in %), (ii) the execution time of our graph compression algorithm, and (iii) the execution time of computing the network identifiability index $\Omega(\mathcal{N})$ (IDs) before and after compressing the graph. We consider that 5% of the non-monitors are randomly selected as monitors and that CAP is used as probing scheme.

7.2.1 ***Real-World Topologies Results***. Table 4 lists the results for the Rocketfuel dataset. As can be seen, our solution can effectively reduce the number of edges of large and complex networks. For example, the number of edges in LEVEL-3 topology, the largest in this dataset, is reduced in 38.92% (from 5,298 to 3,236 edges). Additionally, the computational overhead of compressing the graph is almost negligible when being compared with the monitor placement and network tomography algorithms. It takes only 0.279 seconds for our approach to compress the largest topology. Furthermore, from the experiments, we also observe that higher graph compression rates lead to shorter execution times when computing the network identifiability, reflecting our intuition discussed in Section 5.

7.2.2 ***Synthetic Topologies Results***. Fig. 6 depicts the execution times and compression rates for applying our graph compression to synthetic topologies. Each data point represents the average and standard deviation (std) of applying our approach to 50 different random graphs. The error bars are in green. As shown in Fig.6 (a)-(c), given networks with the same size, execution time and graph compression rates increase when network degree increases. This is due to our approach being able of better compressing edges between nodes which share a common neighbourhood (i.e., similar paths - Section 5), which is more likely to happen in networks with higher density.

The random graphs generated with ER model show similar trends, but with higher compression rates (between 20 to 30%). This is caused by the binomial distribution of ER, which generates denser

Table 4. Graph Compression Results for Rocketfuel dataset. New Edges: number of edges after compression. New ID Time: computational time of the network identifiability after compression. Comp: Compression Rate (graph and time compression)

Network	Time(s)	Edges	New Edges	Comp. (%)	ID Time (s)	New ID Time (s)	Comp. (%)
ABOVE	0.001	294	275	6.46%	10.328	9.881	4.32%
EBONE	0.004	381	341	10.50%	10.140	10.106	0.34%
TISCALI	0.001	404	376	6.93%	14.619	14.465	1.05%
EXODUS	0.003	434	382	11.98%	13.642	11.938	12.49%
TELSTRA	0.002	758	532	29.82%	21.046	19.801	5.91%
ATT	0.009	2078	1296	37.63%	61.831	55.638	10.12%
SPRINTLINK	0.019	2260	1466	35.13%	61.145	53.287	12.85%
VERIO	0.016	2821	2164	23.29%	113.744	107.981	5.07%
LEVEL3	0.279	5298	3236	38.92%	96.952	76.493	21.10%

networks compared to BA and WS. For example, when generating the graphs consisting 10,000 nodes with ER model, these graphs consist of 2,436,624 edges; while the graphs generated with BA and WS models with the same setting consist of only 59,961 and 80,000 edges, respectively.

Fig.6 (d)-(f) show the results obtained when fixing the network degree and varying the size. In this case, the number of nodes has a significant impact on execution time, with larger networks requiring more time for compression. Since graphs generated with ER models have much more edges compared to the other two models, as previously explained, the execution time for ER is significantly longer (57.8 seconds). Regarding compression rates, the reduction is larger for graphs generated with BA and WS models. This is caused by the scale-free (BA) and small-world (WS) properties, which make neighbouring nodes share more similar paths compared to ER. Additionally, given that BA networks consists of a large number of edges connected to smaller numbers of nodes, it generates fewer local areas with high edge similarity and, consequently, smaller compression ratios compared to WS.

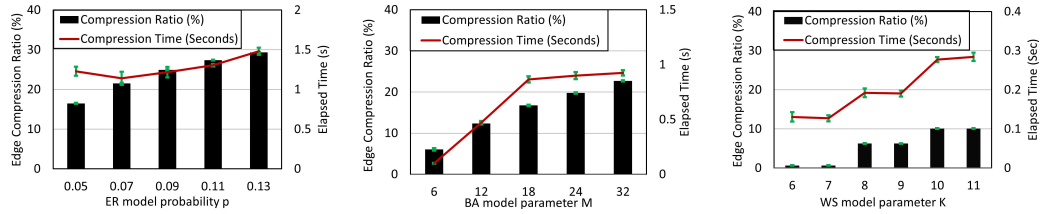
For ER models, the graph compression rates achieve peak when $|V| = 1000$ and reduce in larger topologies. This is due the fact that nodes share less common paths in larger networks, given that edges between nodes have independent probability.

Lastly, it is worth mention that the execution time of our graph compression algorithm remains negligible when being compared to monitor placement algorithms for synthetic topologies.

7.3 Affected Non-Monitors $\mathcal{A}$ and Candidate Monitors C Sets

In Section 6, we argue that given a topology change, it is likely that only a small number of non-monitor nodes $n \in \mathcal{N}$ is affected by this change, i.e., has their node identifiability indexes $\Omega(n)$ reduced. This assumption is the foundation of our adaptive monitor placement (AMP) algorithm. Therefore, in this subsection, we demonstrate how the size of both final affected non-monitors $\mathcal{A}$ and candidate monitors C varies given different topology changes in both real-world and synthetic data sets.

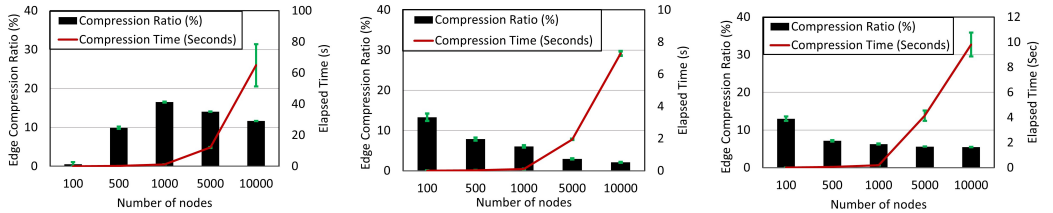
In these experiments, we initialised $\Omega(\mathcal{N})$ by assuming CAP as probing scheme and randomly selecting 5% of the nodes as monitors (initial monitor set). After this initialisation phase, we **randomly remove 1%, 5%, 10%, and 15% of either edges or nodes** from the topologies to simulate link failures (e.g., caused by interference or weather conditions [7]) and devices failures



(a) Different average degrees generated with ER model ($|\mathcal{V}| = 1000$).

(b) Different average degrees generated with BA model ($|\mathcal{V}| = 1000$).

(c) Different average degrees generated with WS model ($|\mathcal{V}| = 1000$).



(d) Different network sizes generated with ER model ($p = 0.05$).

(e) Different network sizes generated with BA model ($M = 6$).

(f) Different network sizes generated with WS model ($K = 8$).

Fig. 6. Graph compression with synthetic random graphs generated with ER, BA and WS models.

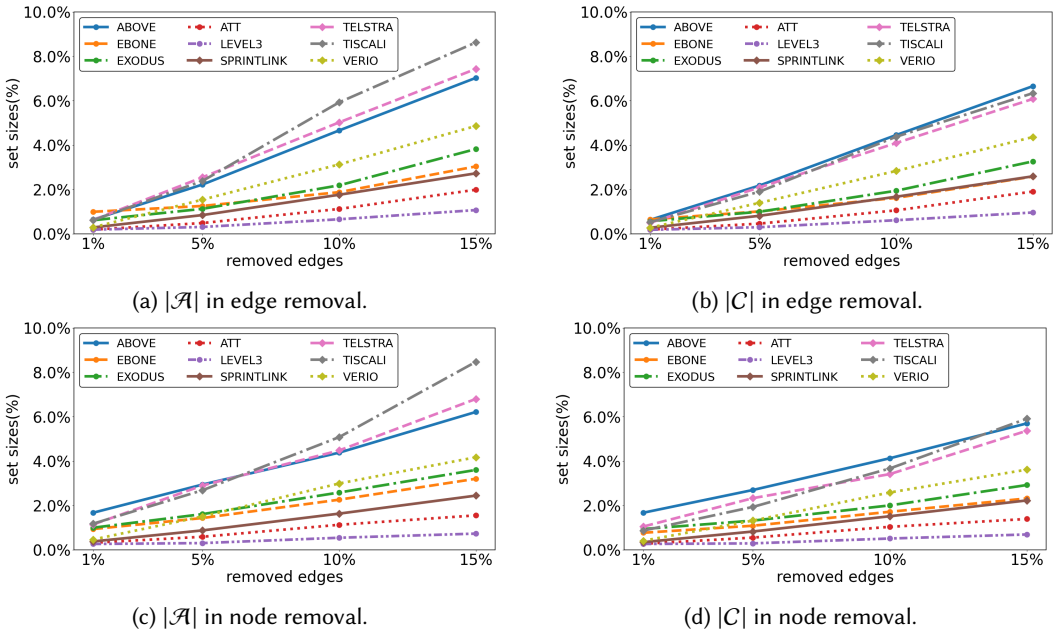
(e.g., caused by hardware issues or malicious attacks [67]), respectively. The obtained results are the average value computed from 100 different runs.

For simplicity, we consider that monitors do not fail or deplete their energy during the node failure localisation process. The topology changes here analysed are, therefore, caused only by non-monitor failures (in the case of node removal). It is worth mentioning, however, that AMP can also be applied in monitor failure scenarios, which can be modelled as a set of edge removals between non-monitors nodes and the virtual monitor m'.

7.3.1 ***Real-World Topologies Results.*** Fig 7 shows the results of edge and node removal for real-world topologies. As can be seen, both $|\mathcal{A}|$ and $|C|$ remain small, i.e., less than 9% of the total number of nodes, under most of the circumstances. Given the same proportion of topology changes, $|\mathcal{A}|$ and $|C|$ is more significant in those topologies having lower degrees. For example, in the scenario where 15% of the edges are removed, $\mathcal{A}$ consist of 6% of nodes in TELSTRA and 5% in TISCALI, which are both sparse networks. LEVEL-3, on the other hand, has the lowest affected set, consisting of only 0.8% of nodes.

Given that all these topologies have similar low connectivity (the value of $\Omega(\mathcal{N})$ is equal 1 in the initialisation phase of all cases), we observe that sparse networks are more vulnerable to topology changes, leading to larger affected areas. It is also worth mentioning that the size of both sets is smaller when considering node removal compared to edge removal. This behaviour is because, when a node is removed, only the edges connected to it are removed from the network, which impacts only the augmenting paths passing through the deleted nodes.

7.3.2 ***Synthetic Topologies Results.*** Figures 8 and 9 depict results obtained for the synthetic networks ER, BA, and WS. Contrary to the real-world topologies, the size of $\mathcal{A}$ and C increases with the degree of the network. However, the network identifiability index $\Omega(\mathcal{N})$ also increases with the degree, meaning that more augmenting paths are created in high degree networks. It is also observed that the size of both sets are equal for all networks, meaning that these topologies generate only **TRIVIAL cases, i.e., cases where the affected non-monitors are only isolated**



(a) $|\mathcal{A}|$ in edge removal.

(b) $|C|$ in edge removal.

(c) $|\mathcal{A}|$ in node removal.

(d) $|C|$ in node removal.

Fig. 7. Size of the affected set $\mathcal{A}$ and candidate set C (in terms of percentage of the number of non-monitor nodes) for different edges removal ratios (Top) and node removal ratios (Bottom) for Rocketfuel topologies

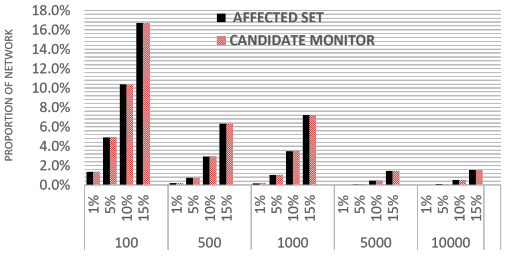
nodes, and the best combination of candidate nodes is the affected nodes set. **TRIVIAL** cases will be further explained in Section 8.6 (London Bus Stop Case Study), but here it is important to understand that the high connectivity and the uniform distribution of the nodes' degree, which are common to synthetic topologies, generate these cases.

Even though these differences between Real-World and Synthetic topologies, it is clear from all the results that the size of both sets is much smaller than the total number of nodes, especially for large-scale networks. These results support our initial hypothesis that the area affected by a topology change is much smaller than the network size, and it is inefficient to search over all non-monitors to adapt a initial monitor placement.

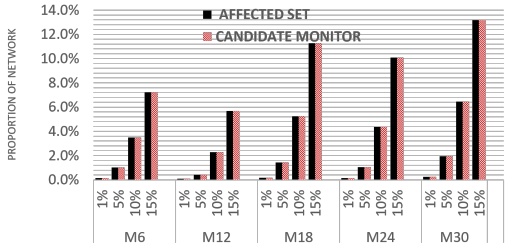
7.4 Performance Analysis of MNMP vs AMP

For this last set of experiments, we compare the execution time of recovering the network identifiability index $\Omega(\mathcal{N})$ using AMP, our proposed adaptive monitor placement, with the state-of-the-art greedy heuristic, MNMP [43]. All the experiments adopted CAP as our probing scheme for simplicity. Similar to the previous experiments, we first randomly select 5% of the nodes as monitors and compute the initial $\Omega(\mathcal{N})$ for this monitor set. Given a topology change, which consist of 10% of the edges or nodes removed for this scenario, AMP adapts this initial monitor set by first compressing the graph and then using the affect non-monitor and candidate monitor sets, $\mathcal{A}$ and C, in the re-computation of $\Omega(\mathcal{N})$. Meanwhile, MNMP uses the original graph and tests all non-monitor nodes when searching for new monitors.

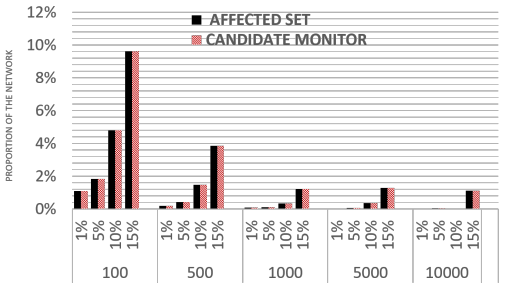
In this set of experiments we only explore the real-world topologies since the synthetic ones only generates **TRIVIAL** cases (Fig. 8 and 9 of Section 7.3 - further discussion in Subsection 8.6). Table 5 shows the obtained results. The first column of each type of topology change (edge or node removal)



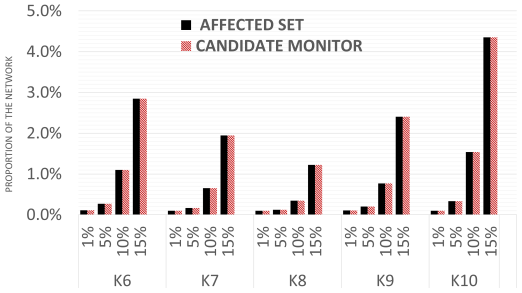
(a) BA model with different network sizes ($M = 6$).



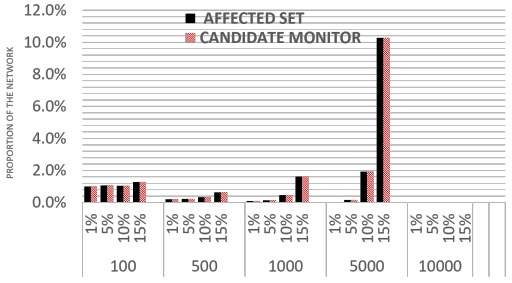
(b) BA model with different average degree ($|\mathcal{V}| = 1000$).



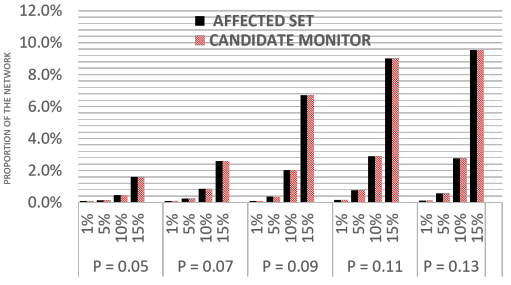
(c) WS model for different network sizes ($K = 8$).



(d) WS model for different network degree ($|\mathcal{V}| = 1000$).



(e) ER model for different network sizes ($P = 0.05$).



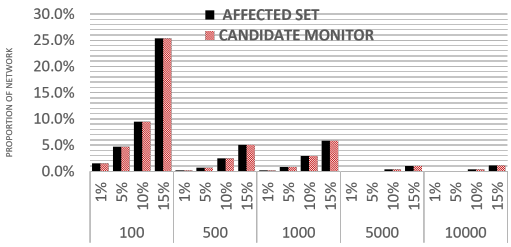
(f) ER model for different network degrees ($|\mathcal{V}| = 1000$).

Fig. 8. Affected non-monitor $\mathcal{A}$ and candidate monitor C sets for different **RANDOM EDGE REMOVAL** ratios (1, 5, 10, and 15%) in synthetic random graphs generated with BA[(a)-(b)], WS[(c)-(d)], and ER[(e)-(f)] models.

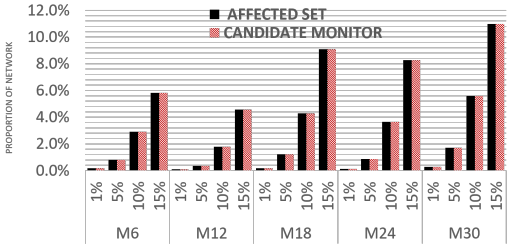
describes the execution time of the state-of-the-art approach MNMP; meanwhile the second and third columns shows the results obtained for AMP with and without graph compression.

As can be seen, the difference in performance is stunning, with AMP taking less than one minute to recover the identifiability levels in most cases, while MNMP takes hours, even days for large networks. This contrast is a consequence of both $\mathcal{A}$ and C being much smaller than the size of the network, as shown in the experiments of previous sections.

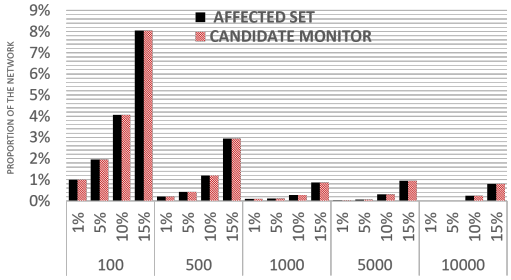
Another example of the impact of $\mathcal{A}$ and C in the performance of AMP is the fact that small networks like TISCALI or TELSTRA have higher execution time compared to bigger networks such as ATT and LEVEL3. As discussed in Section 7.3, sparse real-world topologies have bigger



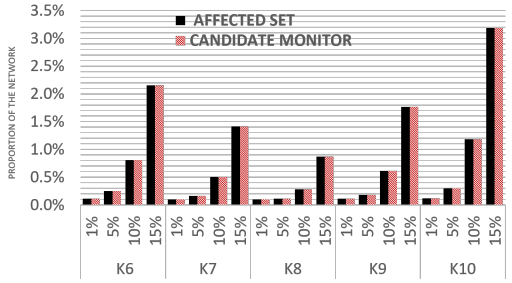
(a) BA model with different network sizes ($M = 6$).



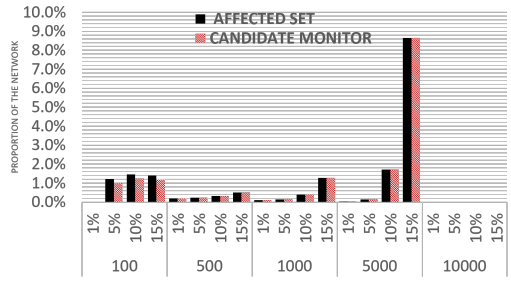
(b) BA model with different average degree ($|\mathcal{V}| = 1000$).



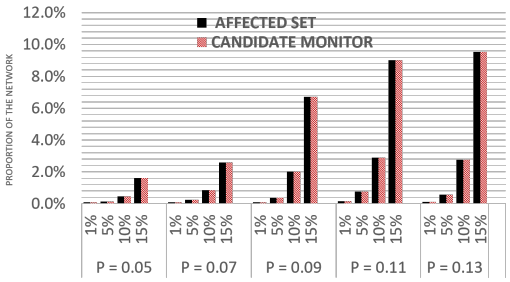
(c) WS model for different network sizes ($K = 8$).



(d) WS model for different network degree ($|\mathcal{V}| = 1000$).



(e) ER model for different network sizes ($P = 0.05$).



(f) ER model for different network degrees ($|\mathcal{V}| = 1000$).

Fig. 9. Affected non-monitor $\mathcal{A}$ and candidate monitor C sets for different **RANDOM NODE REMOVAL** ratios (1, 5, 10 and 15%) in synthetic random graphs generated with BA[(a)-(b)], WS[(c)-(d)], and ER[(e)-(f)] models.

affected areas compared with high-degree ones. Intuitively, the smaller the area affected, the faster the identifiability recovery, independent of the size or degree of the network.

8 CASE STUDY: LONDON BUS-STOP NETWORK

In the previous section, we explored both real-world and synthetic networks to understand the behaviour of our solution under different topologies. In this section, we perform a trace-driven study based on London-Bus Stop topologies [24] to demonstrate the practicality of applying our graph compression and AMP algorithms to real-world city-scale WSN systems. This case study enable us to test the scalability of our solution in non-synthetic topologies, as well as test more topology changes scenarios (details in Section 8.2).

Table 5. Execution time of original MNMP (in clock time format) and of AMP (with and without edge compression) for recovering the network identifiability after removing 10% of the edges and nodes.

	10% EDGE REMOVAL (second)			10% NODE REMOVAL		
Topology	**MNMP**	**AMP**	**AMP (+ compression)**	**MNMP**	**AMP**	**AMP (+ compression)**
ATT	7340.30	4.20	3.23	2732.56	3.26	2.56
LEVEL3	2488.52	2.58	1.77	6059.18	1.78	1.29
SPRINTLINK	12635.62	1.43	1.00	9536.52	1.17	0.82
TISCALI	1806.81	2.59	2.07	1459.51	3.50	3.12
VERIO	112269.10	50.31	48.92	62980.12	47.87	47.99
EXODUS	342.34	0.34	0.21	337.36	0.52	0.39
TELSTRA	3498.61	4.75	3.89	2442.14	7.50	7.39
ABOVE	958.95	1.29	1.23	353.18	2.34	2.21
EBONE	242.56	0.55	0.40	97.13	0.79	0.50

Additionally, a couple of practical observations related to the Initial and Final Affected Sets, $\mathcal{A}_0$ and $\mathcal{A}$, and Candidate Monitor, C, set will be discussed at the end of this Section. The idea is to offer in depth analysis of how these are impacted not only by the level of topology change, but also by the density of different networks.

8.1 Experimental Settings

London-Bus Stop [24], is a data set containing the geographic location of all bus stops around the city of London, UK. This API is made public by Transport for London[2] to stimulate the development of mobile applications for improving the well being and locomotion of citizens around the city. To simulate a Smart City application, we assume a sensing device is attached at each bus stop, forming a mesh topology where each device can communicate with their nearby neighbouring devices. By varying the communication range of these devices, we are able to generate networks with different size and density, as described by Fig. 10 below.

In this case study, we generated 24 different networks from this data set as the following:

(1) We first select two geographical points, St. Paul's Cathedral and the Buckingham Palace, as the centres, and record the position of all the bus stops inside different coverage areas (i.e. 5, 7.5 and 10 km).
(2) We then generate networks by varying the device communication range (i.e. 300, 500, 750 and 1000 meters) of these sensor devices. Note that under the situation that multiple disconnected networks was generated from the same geographical area, we only use the network consisting of the largest number of nodes.

Table 6 describes the network parameters of the 24 networks obtained, while Figure 10 visualises four networks generated when using Buckingham Palace as the geographical centre. For simplicity, in the rest of this section, we use abbreviations to refer different networks. For example, STPAULS R5000 D300 indicates a network using St.Paul Cathedral as geographical centre with coverage area of 5km diameter and communication range 300m.

Similar to the previous experiments, all the algorithms are implemented in MATLAB R2019a and the experiments are performed on a high performance computer (HPC) consisting of 64 cores

[2]https://tfl.gov.uk/

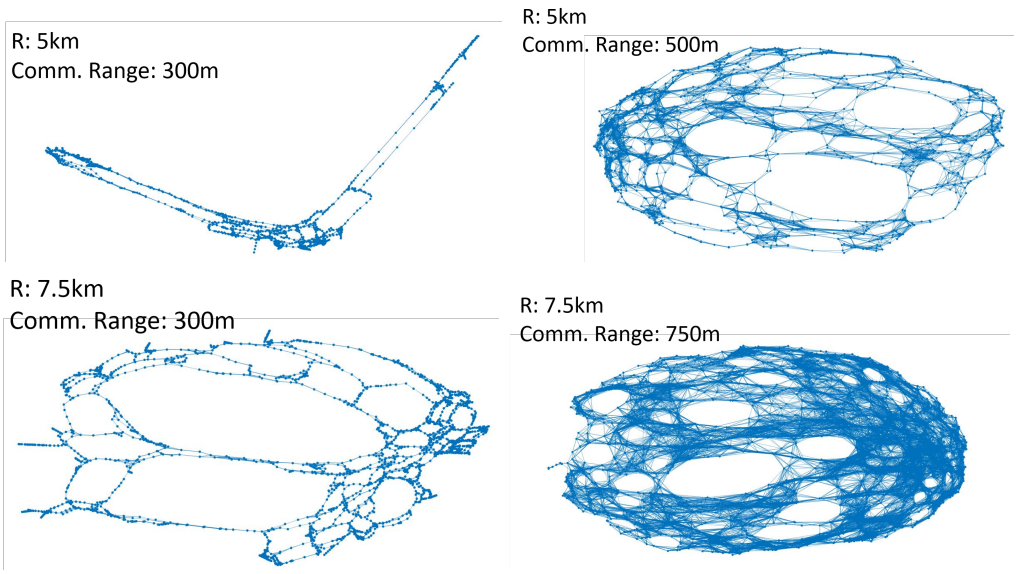


Fig. 10. Examples of networks (given Buckingham palace as geological centre) obtained by varying the radius of coverage (denoted by R) and communication range (denoted by D). We can easily observe that different coverage results in different network sizes (e.g., larger coverage generate larger networks) while communication ranges impact network connectivity (e.g., higher communication range results in denser networks).

mainly composed of dual Xeon E5-series processors, with 500GB of memory. While the experiments were run concurrently, each experiment only exploits one core to reflect the computational power of general PCs.

8.2 Topology Change Model

In this case study, our aim is to test our Graph Compression and Adaptive Monitor Placement (AMP) algorithms against **local and global topology changes that are common to outdoor WSN applications**, such as in Smart Cities. Below we briefly describe the two types of topology changes and how we simulate them in our experiment.

- **Global Topology Changes** are the cases that occur in large scales. One of the most common examples is weather changes where network connectives are impacted by variations in weather conditions (e.g. rain, fog, or snow). Studies show that weather affects the link quality of wireless communication [7, 42, 70], with high temperatures and humidity levels decreasing the Received Signal Strength Indicator (RSSI) of wireless networks. Another example is ageing, where device faults occur due to natural degradation. To simulate these type of topology changes, in each iteration, we randomly remove links and nodes from the network given different proportion (i.e. 1% to 10%).
- **Local Topology Changes** are the cases that only occur within a given geographical area. These changes can be caused by temporal RF-signal blockage (e.g. heavy traffic), communication hot spots [62], faults in the infrastructure, or malicious attacks. To test our approaches under more extreme scenarios, we exploit *jamming attacks* to simulate local topology changes.

Table 6. London Bus Stop Topologies [24] considering St. Paul's Cathedral and Buckingham Palace as the geographic centre points and varying the coverage area (first column) and communication range (second column).

		ST PAUL'S CATHEDRAL				BUCKINGHAM PALACE			
Cover.	**Range**	**Nodes**	**Edges**	**Degree**	**Diameter**	**Nodes**	**Edges**	**Degree**	**Diameter**
5 Km	300 m	1891	8145	8.61	56	1882	8057	8.56	68
	500 m	1956	20115	20.57	25	1906	19447	20.41	29
	750 m	1956	43334	44.31	16	1906	41221	43.25	17
	1000 m	1956	75353	77.5	11	1906	71201	74.71	12
7.5 Km	300 m	3665	14375	7.84	82	3567	14077	7.89	91
	500 m	3794	35666	18.8	40	3704	34424	18.59	40
	750 m	3797	76359	40.22	24	3720	73239	39.38	25
	1000 m	3804	133111	69.98	17	3720	127612	68.61	17
10 Km	300 m	5132	19420	7.57	111	5659	21036	7.43	129
	500 m	6148	54161	17.62	53	6086	52973	17.41	54
	750 m	6173	114642	37.14	32	6093	112647	36.98	33
	1000 m	6173	199046	64.49	23	6093	195831	64.28	23

These attacks consist of generating radio interference to block nearby wireless communications. They are one of the most common denial of service (DoS) attack, being capable of either disrupting the normal system operation or causing performance degradation by keeping the communication channel busy [29, 69]. To simulate these jamming attacks, in each iteration, a jamming coverage area is defined (i.e. 300, 500, 750, and 1000m) given a random geographical location. All nodes inside the area are considered unresponsive (i.e. their communication links are removed) during the attack. This is similar to the cases of node removals, but taking spatial-correlation into account.

8.3 Experiment Results of Graph Compression $\mathcal{G} \rightarrow \hat{\mathcal{G}}$

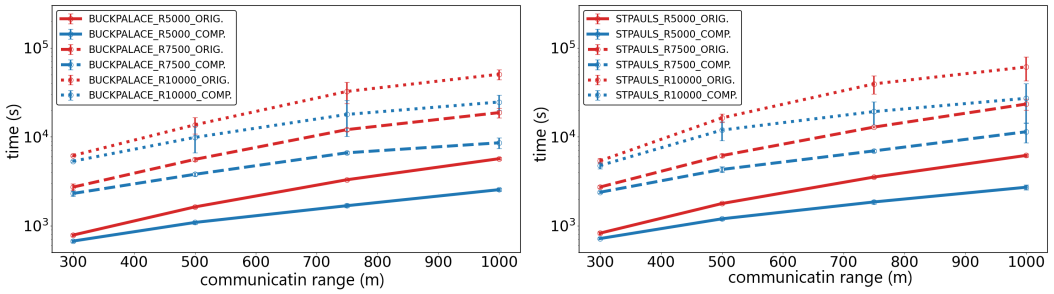
In this subsection, we tested the graph compression algorithm proposed in Section 5. Table 7 shows the compression results obtained from St. Paul's and Buckingham Palace networks. Similar to our evaluation in Section 7.2, execution times of the compression algorithm are negligible compared to the monitor placement algorithm. It only took 3.454 seconds to compress the largest network (i.e. STPAULS R10000) consisting of 6,173 nodes and 199,046 edges. A clear positive correlation is also observed between number of edges and compression time.

The results also show higher compression rates compared to both Real-World and Synthetic topologies previously explored in Subsection 7.2. Our solution reduces between 23.16% - 60.52% and 22.75% - 60.41% of the number of edges given networks centred St Paul's Cathedral and Buckingham Palace, respectively. This is due to the fact that in WSNs, geographically neighbouring nodes typically share similarities in the communication paths used. This property matches our graph-compression algorithm and therefore results in higher compression rates. It is worth noting that the compression rates only depend on average degree and are independent of the number of nodes.

Table 7. Graph Compression Results for St. Paul's Cathedral and Buckingham Palace. Original and Comp. consist of the number of edges before and after the compression, respectively, and rate is the edge compression rate.

		ST PAUL'S CATHEDRAL				BUCKINGHAM PALACE			
Cover.	**Range**	**Time (s)**	**Original**	**Comp.**	**Rate (%)**	**Time (s)**	**Original**	**Comp.**	**Rate (%)**
5 Km	300 m	0.181	8145	6004	26.29%	0.187	8057	5903	26.73%
	500 m	0.699	20115	11219	44.23%	0.698	19447	10843	44.24%
	750 m	0.984	43334	19323	55.41%	0.877	41221	18276	55.66%
	1000 m	1.336	75353	29752	60.52%	1.336	71201	28190	60.41%
7.5 Km	300 m	0.569	14375	10961	23.75%	0.538	14077	10642	24.40%
	500 m	1.512	35666	20663	42.07%	1.42	34424	19894	42.21%
	750 m	1.688	76359	35022	54.14%	1.718	73239	33592	54.13%
	1000 m	2.248	133111	54828	58.81%	2.152	127612	52316	59.00%
10 Km	300 m	1.001	19420	14922	23.16%	1.167	21036	16244	22.78%
	500 m	2.361	54161	31966	40.98%	2.531	52984	31434	40.67%
	750 m	2.789	114642	53765	53.10%	2.699	112647	52992	52.96%
	1000 m	3.454	199046	83068	58.27%	3.174	195831	82037	58.11%

Figure 11 depicts the time required to the compute network identifiability index $\Omega(\mathcal{N})$ given a node randomly selected as a monitor and CAP as probing scheme. Each data point is collected from 10 different experiments. As can be seen, the gain in execution time between original and compressed graph grows with the average network degree (indicated by the communication range), following the graph compression trends observed in Table 7. When communication ranges $\geq$ 750m, the computation of $\Omega(\mathcal{N})$ it is at least 2x faster when adopting our graph compression scheme.



(a) Buckingham Palace

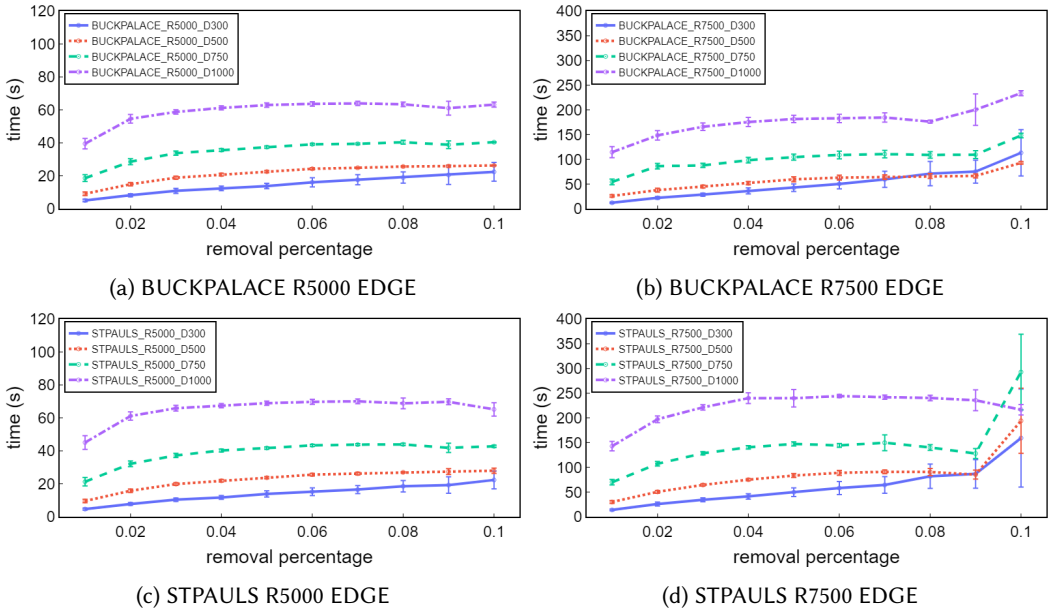
(b) St. Paul's Cathedral

Fig. 11. Execution time for obtaining the network identifiability $\Omega(\mathcal{N})$ in the original graph (red lines) and compressed graph (blue lines) for Buckingham Palace (Fig. a) and St. Paul's (Fig. b) topologies. Different line styles define different coverage radius (described by RX, where X = 5, 7.5, and 10Km), while the x axis describe the communication range. The higher the network's density, the higher the execution time gain offered by the graph compression algorithm

8.4 Experiment Results of Adaptive Monitor Placement

In this subsection, we demonstrate the computational overhead of AMP against the Global and Local topology changes previously described (Section 8.2) while adopting CAP as our probing scheme. Here, AMP first tries to identify if the network identifiability index $\Omega(\mathcal{N})$ is reduced. If yes, it tries to recover $\Omega(\mathcal{N})$ by using $\mathcal{A}$ and C to select new monitors.

8.4.1 ***Random Edge Removal.*** In this set of experiments, we randomly removed 1% to 10% of edges given the compressed graphs of the described networks to simulate the impacts due to changes in physical environments, including weather [49] and traffic conditions [62]. Results are depicted in Fig. 12, where the number behind R and D indicate a network coverage (e.g. R5000 denotes 5,000 meters from centre location) and communication range (e.g. D300 denotes link exist if distances between nodes are below 300 meter), respectively. As can bee seen, AMP is able to adapt to these topology changes within 60s and 250s given that 10% of the edges are removed from R5000 and R7500 networks, respectively. The computational overhead increases in a sub-linear manner with respect to the number of edge removal. Exceptions are observed in the cases where 10% of edges are removed in Fig.12 (b) and (d). The average computational time and variation are much higher compared to remaining results. This is because the edges with higher weights in the compressed graph were removed, causing significant interruption in available paths for network tomography. In these cases, AMP requires the selection of a significant amount of new monitors to recover $\Omega(\mathcal{N})$ to its original states. Higher variations were also observed in these cases.



(a) BUCKPALACE R5000 EDGE (b) BUCKPALACE R7500 EDGE

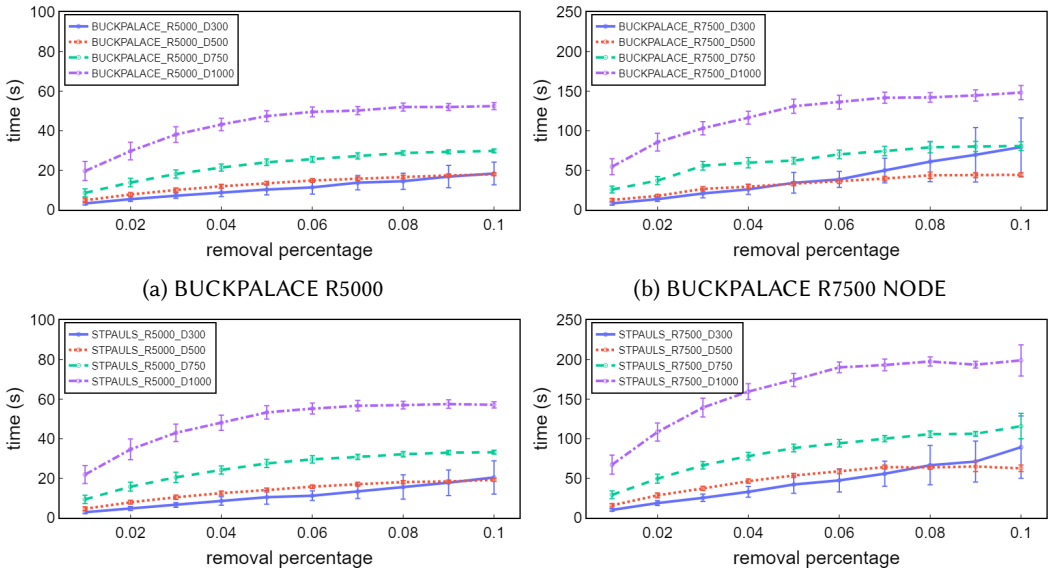
(c) STPAULS R5000 EDGE (d) STPAULS R7500 EDGE

Fig. 12. Time required for AMP to adapt to topology changes given random edge removals (from 1% to 10%) in compressed graph.

8.4.2 ***Random Node Removal.*** Similar to edge removal, here we randomly removed 1% to 10% of non-monitor nodes from the studied networks to simulate device failures (caused by low power, damages or program bugs) [51] that are often observed in WSNs from moderate to severe situations. It is possible to observe that the results depicted in Fig. 13 are similar to edge removal while being

10%-15% faster. After investigating the results, we realised that fewer edges were actually removed compared to edge removal cases, that is, 10% of node removal results in less then 10% of edge removed. This leads to less paths between non-monitor and monitors being removed, making the recovery of $\Omega(\mathcal{N})$ faster.

Once again, higher variation and computational overhead are observed in Fig.13 (b) and (d) when a higher percentage of nodes are removed. This was caused by our node removal scheme where dummy nodes traversed by edges with higher weights were likely to be removed given higher percentage of node removal.



(a) BUCKPALACE R5000

(b) BUCKPALACE R7500 NODE

(c) STPAULS R5000 NODE

(d) STPAULS R7500 NODE

Fig. 13. Time required for AMP to adapt to topology changes given random node removals (from 1% to 10%).

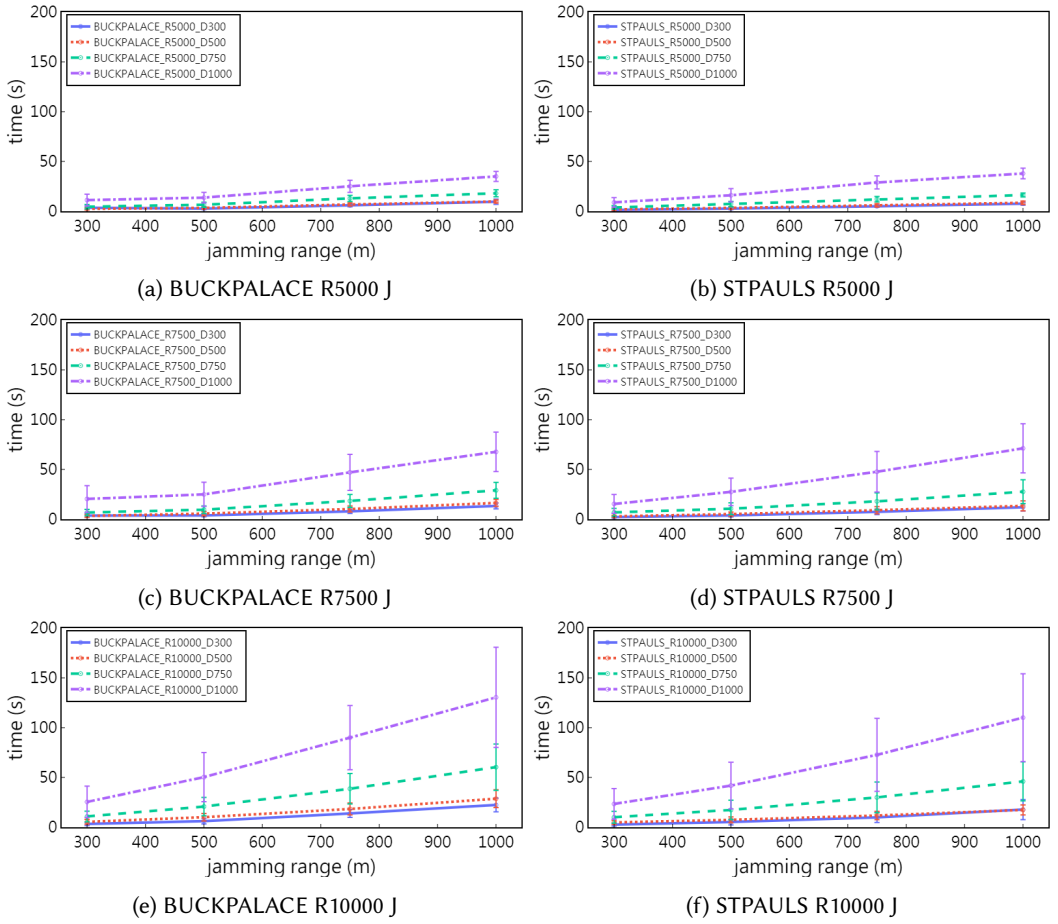
8.5 Jamming Attack

In the Jamming Attack experiments, we randomly add a jammer, which is able to interfere with nearby network links, in each case analysed. We consider that the interference range varies between 300 to 1,000 meters (i.e. D300 to D1000). This behaviour is simulated by removing all the edges (i.e. network links) within the different ranges of interference. Results are depicted in Fig. 14 As indicated in the results, AMP can efficiently cope with these topology changes within 128 seconds given the largest network (i.e. BUCKPALACE R10000 D1000) and interference range (i.e. 1,000 meters). In most of the cases, where interference ranges are smaller, AMP can cope with these changes within 10 seconds.

8.6 Observation and Discussion

As observed in the previous experiments, AMP shows great potential to cope with the dynamic changes in real-world large-scale scenarios. In this subsection, we would like to discuss a couple of observations obtained during the experiments in relation to the affected and candidate monitor sets, providing guidelines to users who would like to implement AMP in real-world applications.

As demonstrated in Section 6, AMP consists of four different steps:



(a) BUCKPALACE R5000 J (b) STPAULS R5000 J

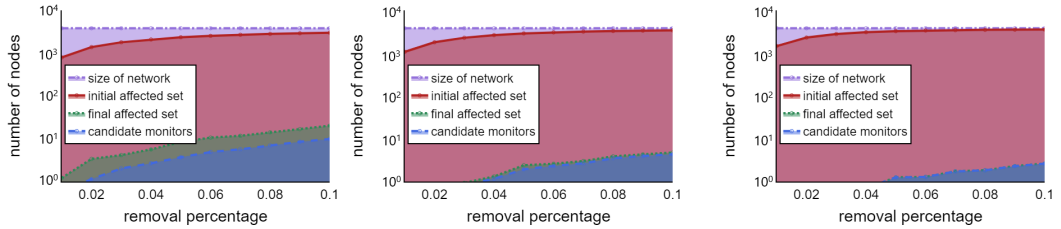
(c) BUCKPALACE R7500 J (d) STPAULS R7500 J

(e) BUCKPALACE R10000 J (f) STPAULS R10000 J

Fig. 14. Time required for AMP to adapt to topology changes given jamming attacks with different interference ranges (from 300 to 1,000 meters).

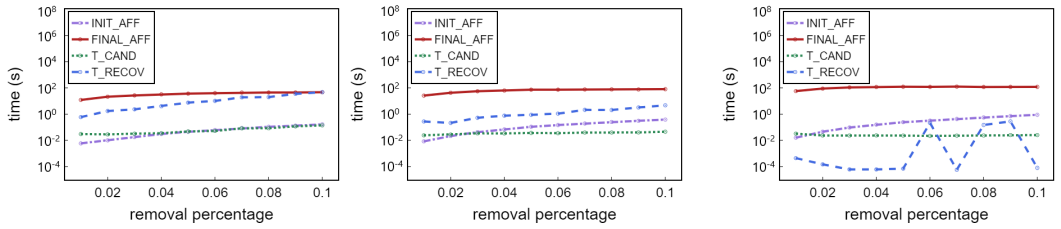
(1) **Compute initial affected set $\mathcal{A}_0$.** AMP first identifies the nodes $n \in \mathcal{N}$ whose identifiability indexes $\Omega(n)$ are affected by any topology change (i.e., edge and node removals). This yields a computational complexity of $O(|\Delta\hat{\mathcal{E}}||\mathcal{A}_0|)$ (where $|\Delta\hat{\mathcal{E}}|$ denotes the number of edge removed).
(2) **Compute final affected set $\mathcal{A}$.** Once this *initial affected set* $\mathcal{A}_0$ is acquired, AMP reduces $\mathcal{A}_0$ to a final affected set $\mathcal{A}$ by keeping only the non-monitors whose identifiability indexes becomes less than k ($\Omega(n) < k$). This yields $O(|\hat{\mathcal{E}}||\mathcal{N}||\mathcal{A}_0|)$ computational complexity.
(3) **Compute candidate monitor set C.** With $\mathcal{A}$, AMP find candidate monitors $C \subset \mathcal{N}$, yielding a computational complexity of $O(|\hat{\mathcal{E}}||\mathcal{N}||\mathcal{A}|)$.
(4) **Select monitor from C to recover $\Omega(\mathcal{A})$.** Ultimately, AMP selects monitors $m \in C$ to recover $\Omega(n)$ for every $n \in \mathcal{A}$ to the required level k. This yields $O(|\hat{\mathcal{E}}|^2|\mathcal{N}||\mathcal{A}||C|)$ computational complexity.

Fig.15 illustrates the size of these four sets given different percentage of edge removal. As can be seen, the size of the initial affected set $\mathcal{A}_0$ is close to the size of the non-monitor set $\mathcal{N}$ in denser networks (e.g. D500 and D750) with higher edge removal percentages. In contrast, final affected sets



(a) STPAULS R7500 D300 E (b) STPAULS R7500 D500 E (c) STPAULS R7500 D750 E

Fig. 15. Different set sizes in the edge removal experiment with STPAULS R7500 network.



(a) STPAULS R7500 D300 E (b) STPAULS R7500 D500 E (c) STPAULS R7500 D750 E

Fig. 16. Computational time of each set in the edge removal experiment with STPAULS R7500 network E.

$\mathcal{A}$ and candidate monitors sets C are 2.5 and 3 degrees of magnitude smaller than $\mathcal{N}$, respectively. Also, the sizes of both these sets reduce in denser networks.

To better understand the overhead of each step, we further extracted the computational time of each set reduction from the experiment results. Fig.16 depicts examples of our edge removal experiments with STPAUL R7500 network given different communication range between 300m to 750m. As can be seen, the initial affected set $\mathcal{A}_0$ (purple lines) can be acquired in between 0.01 and 1 second for all cases; final affected sets $\mathcal{A}$ (red lines) require 12-120 seconds to be computed; and candidate monitors C (green lines) can be obtained in around 0.05 seconds. All of the above operations are less sensitive to changes in the edge removal percentages (at most linear growth).

In contrast, different network densities yield distinctively different computational overhead when recovering network identifiability index (blue lines). In the low density network ($D = 300$), the average recovery time is 0.6-47.5 seconds, which is similar to $\mathcal{A}$; when $D = 500$, the computational overhead grows to another order of magnitude, varying between 0.27-4.7 seconds, which is comparable to $\mathcal{A}_0$ and C. Finally, in the densest of the networks ($D = 750$), the computational overhead is significantly lower then other operations, given by 60μ-0.28 seconds with high variation (larger than 2 orders of magnitude).

After further investigating our raw experimental results, we understand this variation is caused by the three distinctive cases where AMP behaves differently. These cases are described bellow:

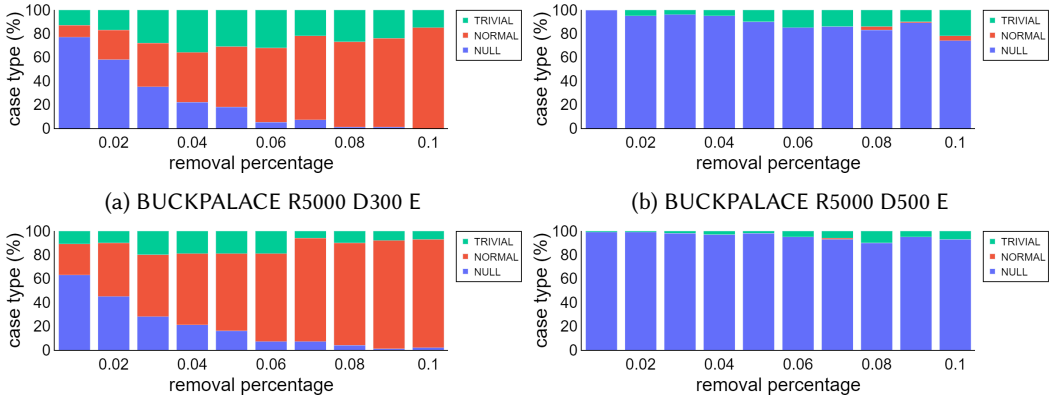
- **NORMAL cases ($|\mathcal{A}| > |C|$)** are the situations where network identifiability index is below threshold k (i.e. $\Omega(\mathcal{N}) < k$) and require recovery. Under this situation, AMP is able to find a small set of candidates C from which a subset of nodes are selected as new monitors.
- **TRIVIAL cases ($|\mathcal{A}| = |C|$)** are the situations where exist no better way to recover $\Omega(\mathcal{N})$ to k unless choosing all nodes in the affected set $\mathcal{A}$ as monitors. This situations occur when there exists no node $n \in \mathcal{N}$ which can simultaneously recover k for multiple affected nodes

in $\mathcal{A}$ and the virtual monitor m' in the auxiliary graph. The search for the best candidate monitors results in the same nodes in $\mathcal{A}$.

- **NULL cases ($\mathcal{A} = \emptyset$)** are the situation where $\Omega(\mathcal{N}) \geq k$ given the topology changes. Since no new monitor is required, AMP does not search for candidate set but simply updates its *inverted index of visited edges* $\mathbf{L}^{-1}$ accordingly.

Fig.17 and 18 illustrate the distribution of these three cases in our experiments. We observe that **NULL** cases are likely to appear in dense networks with smaller percentage of elements removal. For example, our experimental results show that given the node removal in both networks with parameter R=7500 and D=500, **NULL** cases are predominant when removal percentage is smaller than 5%. In contrast, **NORMAL** cases are more frequently observed in low density networks with higher percentage of removal. **TRIVIAL** cases appear as transit in between null and normal cases.

The case distribution in Fig.18 (a) and (b) clearly explains the variation observed in Fig. 16. By comparing Fig. 16 (a) and Fig.18 (a), we noticed that the number of **NORMAL** cases has direct impact on the computational complexity in the last step of AMP (i.e., selecting monitors from C to recover $\Omega(\mathcal{A})$). Similar trend is observed in Fig. 16 (b) where most of the cases are dominant by **NULL** and **TRIVIAL** cases.



(a) BUCKPALACE R5000 D300 E

(b) BUCKPALACE R5000 D500 E

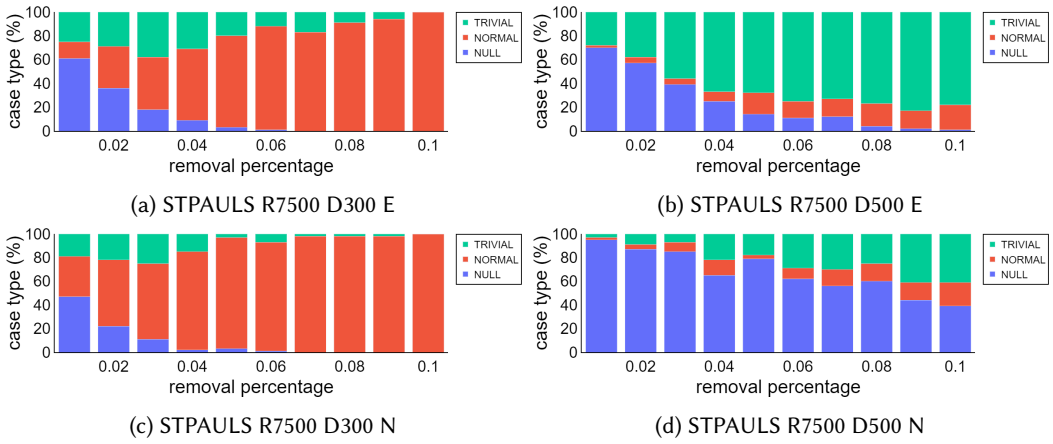
(c) BUCKPALACE R5000 D300 N

(d) BUCKPALACE R5000 D500 N

Fig. 17. Case distribution at BUCKPALACE.

These results led us to conclude that there is a high correlation between the network density and its capability in adapting the initial monitor set given topology changes. Intuitively, non-monitors have higher identifiability indexes in denser networks due to the large number of augmenting paths between non-monitors and the virtual monitor m' in the auxiliary graphs (explained in Sections 3.5 and 3.6). Additionally, this large number of paths allows the non-monitors to use alternative paths in the re-computation of $\Omega(n)$ after a topology change, resulting in many of the **NULL** cases observed.

To validate this intuition and better understand the causes behind each of theses cases, we further investigate the distribution of individual node identifiability indexes for different network densities. The results are plotted in Fig.19. As can be seen, the distribution of non-monitors identifiability indexes $\Omega(n)$, with $n \in \mathcal{N}$, has smaller mean and variance in lower density networks (e.g. $D = 300$ in our experiment) due to the limited number of augmenting paths. Edge and node removals, therefore, have higher probability to reduce $\Omega(n)$ to the level below the original network identifiability index (i.e. the smallest value of $\Omega(n)$) that we aim to recover, resulting in larger sizes of $\mathcal{A}$, as illustrated

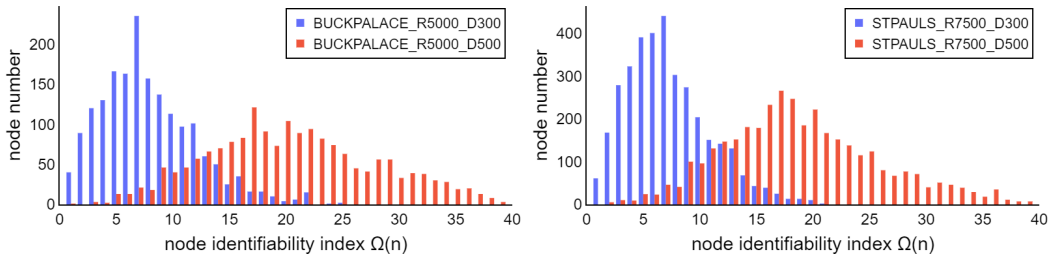


(a) STPAULS R7500 D300 E (b) STPAULS R7500 D500 E

(c) STPAULS R7500 D300 N (d) STPAULS R7500 D500 N

Fig. 18. Case distribution at STPAULS.

in Fig.15. In contrast, in denser networks (e.g. D=500 in our experiment), the $\mathcal{A}$ is typically smaller due to larger mean and variance in the distribution of $\Omega(n)$, which ultimately results in higher proportion of **NULL** and **TRIVIAL** cases, given that the identifiability indexes are less impacted by changes in topology.



(a) BUCKPALACE R5000 (b) STPAULS R7500

Fig. 19. Distribution of non-monitors identifiability indexes $\Omega(n)$.

It is worth noticing that **TRIVIAL** cases were frequently observed in the synthetic topologies studied in Section 7.3. Synthetic topologies usually have a more uniform distribution of identifiability indexes, since their density follow a mathematical formulation. Given that, they present a similar behaviour of the dense London Bus-Stop Networks observed in Fig. 19. By contrast, Real-world networks, such as Rocketfuel, have a more random distribution of $\Omega(n)$, which explain the number of **NORMAL** cases observed.

9 CONCLUSION AND FUTURE WORK

In this paper we explore two methods for efficiently solving the Monitor Placement problem in Large-Scale WSNs: **Graph Compression** and **Adaptive Monitor Placement (AMP)**. The main goal of these two methods is to find the optimum monitor set $\mathcal{M}$ in a faster and adaptive way, enabling the monitor placement to adjust to topology changes while maintaining the required value of network identifiability index $\Omega(\mathcal{N})$. Since this index measures the capability of a network

in localising multiple node failures using Network Tomography, it is desirable to maintain its value above a defined threshold k for long periods of time and under different topology changes.

Our proposed graph compression algorithm (Section 5) improves the computational time of $\Omega(\mathcal{N})$ by exploring the similarities between neighbouring nodes to reduce the number of edges. An edge weight assignment algorithm is also developed to guarantee that the maximum flow of the weighted compressed graph $\hat{\mathcal{G}}$ is equal to the maximum flow of the original graph, obtaining the same network identifiability index for both graphs. Our experimental analysis on both real and synthetic networks shows that this method enables graph compression rates of up to 60% in large and dense networks, reducing the network identifiability index computational time by at most 50% with a negligible pre-processing time.

On the other hand, our proposed Adaptive Monitor Placement method (AMP) (Section 6) reduces the search of new monitors for efficiently recovering the value of $\Omega(\mathcal{N})$ given variations in topology. Firstly, AMP obtain the affected non-monitor set $\mathcal{A}$, a subset of non-monitors whose identifiability index $\Omega(n)$ were affected by the topology change (i.e., a list of edges or nodes removed from the network). Secondly, AMP search for the candidate monitor set C, a subset of non-monitors whose selection as monitor will contribute in the recovery of the identifiability indexes of the nodes in $\mathcal{A}$. Given this, it is more efficient to consider only these sets in the re-computation of the $\Omega(\mathcal{N})$ instead of searching through all non-monitor for the best candidate monitors, as proposed by the state-of-the-art solutions.

Extensive experiments with different topologies showed that the sizes of $\mathcal{A}$ and C are much smaller than the size of the network, even for high level topology changes (e.g, 10% of edges removed), with both sets being impacted by the connectivity of the network, not the scale (discussion on Section 8.6). Additionally, by using both sets, AMP managed to recover $\Omega(\mathcal{N})$ in less than 1 minute for most cases, while the state-of-the-art approaches take hours, or even days for larger networks.

Finally, in a Case Study using the London Bus-Stop API (Section 8), we showed the practicality of applying our solutions for node failure localisation in Smart City applications. The experiments showed that, even for large-scale networks with more than 1,000 nodes, Graph Compression + AMP is capable of adjusting the monitor placement in less than 100 seconds for most tested cases, even for malicious attacks scenarios such as Jamming.

9.1 A discussion on value of k and the future of AMP

In network tomography, the parameter k defines the number of uniquely identifiable faults, i.e., the number of anomalous nodes the monitor placement can identify (details in Section 3). The value of k is usually defined by developers and users based on the system's reliability requirements, with higher k resulting in a higher capability of the system in localising node failures. However, recent work in network identifiability shows that the network topology imposes some limitations on the maximum k achievable [10]. The authors of [44], for example, observed that sparse topologies (e.g., tree or grid) require more monitors than dense ones (mesh) for achieving the same value of k.

Given this, it is challenging to use the value of k to compare different failure scenarios since different topologies may achieve different values of k for the same number of monitors and vice-versa. Additionally, considering that our main goal as an adaptive approach is to observe variations in topologies and how these variations impact the system's capability in detecting failures, we are more interested in observing how k varies (Δk) for different topologies given (1) an initial monitor placement and (2) different failure scenarios.

Therefore, when modelling our solution AMP, we considered a fixed k defined at deployment time based on an initial monitor placement. This k is used as a threshold level representing the

system's reliability requirements. If this value decreases, AMP can quickly re-establish the initial k by efficiently finding the areas affected and selecting more monitors. If the threshold value of k changes during the system life cycle due to, for example, changes in the application's reliability requirement instead of topology changes, AMP will not be able to adjust the monitor placement correctly. To keep AMP working correctly, we need to reset its parameters to the new threshold value k' and then redeploy it in the system. As future work, we need to make AMP intelligent enough be to be able to identify if variations in k are caused by topology or application requirement changes, adapting the monitor placement only when necessary.

It is also worth noting that, in this paper, we only studied cases where the value of k decreases due to topology changes caused by a decrease in connectivity (edges and nodes being removed from the network). We focused on those cases since our goal is to develop methods for maintaining a networked sensor-based system reliable (i.e., still capable of identifying failures) under situations of failures/malicious attacks. However, the ideas used to obtain the affected areas and the candidate monitor set can be adapted to solve other types of topology changes, such as, for example, increased network connectivity due to changes in the radio antennas. As future work, we want to make AMP robust enough to adapt to a wide range of possible topology changes in WSN applications. Preferably, we would like AMP to be able to learn the most common changes in a specific system and improve its performance over time by predicting the possible affected areas before any change occurs.

REFERENCES

[1] Mohammad Aazam, Sherali Zeadally, and Khaled A. Harras. 2018. Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Trans. Ind. Informat.* 14, 10 (2018), 4674–4682.

[2] Nurzaman Ahmed, Debashis De, and Iftekhar Hussain. 2018. Internet of Things (IoT) for smart precision agriculture and farming in rural areas. *IEEE Internet Things J.* 5, 6 (2018), 4890–4899.

[3] Satyajeet S Ahuja, Srinivasan Ramasubramanian, and Marwan Krunz. 2008. SRLG failure localization in all-optical networks using monitoring cycles and paths. In *Proc. IEEE INFOCOM*. IEEE, Phoenix,US, 700–708.

[4] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74, 1 (2002), 47.

[5] Daniel Alonso-Román and Baltasar Beferull-Lozano. 2016. Adaptive consensus-based distributed detection in WSN with unreliable links. In *Proc. IEEE ICASSP*. IEEE, Shanghai, China, 4438–4442.

[6] César Asensio-Marco, Daniel Alonso-Román, and Baltasar Beferull-Lozano. 2018. Cross-layer MAC protocol for unbiased average consensus under random interference. *IEEE Trans. Signal Inf. Process. Netw.* 5, 2 (2018), 320–333.

[7] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. 2012. Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sens. Netw.* 8, 4 (2012), 34.

[8] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM, New York.

[9] Novella Bartolini, Ting He, Viviana Arrigoni, Annalisa Massini, and Hana Khamfroush. 2019. On Fundamental Bounds of Failure Identifiability by Boolean Network Tomography. *CoRR* abs/1903.10636 (2019). arXiv:1903.10636

[10] Novella Bartolini, Ting He, and Hana Khamfroush. 2017. Fundamental limits of failure identifiability by boolean network tomography. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[11] Yigal Bejerano and Rajeev Rastogi. 2006. Robust monitoring of link delays and faults in IP networks. *IEEE/ACM Trans. Netw.* 14, 5 (2006), 1092–1103.

[12] Gregory Buehrer and Kumar Chellapilla. 2008. A scalable pattern mining approach to web graph compression with communities. In *Proc. of WSDM*. ACM, New York, 95–106.

[13] Prasenjit Chanak and Indrajit Banerjee. 2016. Fuzzy rule-based faulty node classification and management scheme for large scale wireless sensor networks. *Expert Syst. Appl.* 45 (2016), 307–321.

[14] Prasenjit Chanak, Indrajit Banerjee, and R Simon Sherratt. 2016. Mobile sink based fault diagnosis scheme for wireless sensor networks. *J. Syst. Softw.* 119 (2016), 45–57.

[15] Sangman Cho and Srinivasan Ramasubramanian. 2014. Localizing link failures in all-optical networks using monitoring tours. *Comput. Networks* 58 (2014), 2–12.

[16] Francisco Claude and Gonzalo Navarro. 2010. Fast and compact web graph representations. *ACM Trans. Web* 4, 4 (2010), 16.

[17] Douglas S.J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. 2003. A high-throughput path metric for multi-hop wireless routing. In *Proc. MobiCom*. ACM, San Diego, CA, 134–146.

[18] Reinhard Diestel. 2018. *Graph theory*. Springer, Berlin.

[19] Nick Duffield. 2003. Simple network performance tomography. In *Proc. ACM SIGCOMM*. ACM, Miami Beach,FL, 210–215.

[20] Nick Duffield. 2006. Network tomography of binary network performance characteristics. *IEEE Trans. Inf. Theory* 52, 12 (2006), 5373–5388.

[21] Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* 19, 2 (1972), 248–264.

[22] Paul Erdős and Alfréd Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.

[23] Glauco Feltrin, Nemanja Popovic, and Michał Wojtera. 2019. A Sentinel Node for Event-Driven Structural Monitoring of Road Bridges Using Wireless Sensor Networks. *Journal of Sensors* 2019 (2019), 1–16.

[24] Transport for London. 2019. Transport for London API. https://api-portal.tfl.gov.uk/docs.

[25] Lester R Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Can. J. Math.* 8, 3 (1956), 399–404.

[26] Yi Gao, Wei Dong, Chun Chen, Xiaoyu Zhang, Jiajun Bu, and Xue Liu. 2016. Accurate per-packet delay tomography in wireless ad hoc networks. *IEEE/ACM Trans. Netw.* 25, 1 (2016), 480–491.

[27] Denisa Ghita, Can Karakus, Katerina Argyraki, and Patrick Thiran. 2011. Shifting network tomography toward a practical goal. In *Proc. of CoNEXT*. ACM, Tokyo, Japan, 24.

[28] Sonam Goyal and Trilok Chand. 2017. Improved trickle algorithm for routing protocol for low power and lossy networks. *IEEE Sensors J.* 18, 5 (2017), 2178–2183.

[29] Kanika Grover, Alvin Lim, and Qing Yang. 2014. Jamming and anti-jamming techniques in wireless networks: a survey. *Int. J. Ad Hoc Ubiquitous Comput.* 17, 4 (2014), 197–215.

[30] Sachin S. Kamble, Angappa Gunasekaran, and Shradha A. Gawankar. 2018. Sustainable Industry 4.0 framework: A systematic literature review identifying the current trends and future perspectives. *Process Saf. and Environ. Prot.* 117 (2018), 408–425.

[31] Fatma Karray, Alberto Garcia-Ortiz, Mohamed W Jmal, Abdulfattah M Obeid, and Mohamed Abid. 2016. Earnpipe: a testbed for smart water pipeline monitoring using wireless sensor network. *Procedia Computer Science* 96 (2016), 285–294.

[32] Matthias Keller, Jan Beutel, and Lothar Thiele. 2012. How was your journey? Uncovering routing dynamics in deployed sensor networks with multi-hop network tomography. In *Proc. of SenSys*. ACM, Toronto, Canada, 15–28.

[33] Abhishek Khanna and Sanmeet Kaur. 2019. Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture. *Comput. Electron. Agric.* 157 (2019), 218–231.

[34] Harith Kharrufa, Hayder AA Al-Kashoash, and Andrew H Kemp. 2019. RPL-based routing protocols in IoT applications: A Review. *IEEE Sensors J.* 19, 15 (2019), 5952–5967.

[35] Xuemin Lin. 2000. On the computational complexity of edge concentration. *Discrete Appl. Math.* 101, 1-3 (2000), 197–205.

[36] Xin Liu, Mai Abdelhakim, Prashant Krishnamurthy, and David Tipper. 2018. Identifying malicious nodes in multihop iot networks using diversity and unsupervised learning. In *Proc. of ICC*. IEEE, Kansas City, US, 1–6.

[37] Yunhao Liu, Yuan He, Mo Li, Jiliang Wang, Kebin Liu, and Xiangyang Li. 2012. Does wireless sensor network scale? A measurement study on GreenOrbs. *IEEE Trans. Parallel Distrib. Syst.* 24, 10 (2012), 1983–1993.

[38] Yunhao Liu, Kebin Liu, and Mo Li. 2010. Passive diagnosis for wireless sensor networks. *IEEE/ACM Trans. Netw.* 18, 4 (2010), 1132–1144.

[39] Yuxin Liu, Ming Ma, Xiao Liu, Naixue Xiong, Anfeng Liu, and Ying Zhu. 2018. Design and analysis of probing route to defense sink-hole attacks for Internet of Things security. *IEEE Trans. Netw. Sci. Eng.* 7 (2018), 356–372.

[40] Chun Lo, Jerome P Lynch, and Mingyan Liu. 2016. Distributed model-based nonlinear sensor fault diagnosis in wireless sensor networks. *Mech. Sys. Sig. Process.* 66 (2016), 470–484.

[41] Weidang Lu, Yi Gong, Xin Liu, Jiaying Wu, and Hong Peng. 2017. Collaborative energy and information transfer in green wireless sensor networks for smart cities. *IEEE Trans. Ind. Informat.* 14, 4 (2017), 1585–1593.

[42] Jari Luomala and Ismo Hakala. 2015. Effects of temperature and humidity on radio signal strength in outdoor wireless sensor networks. In *Proc. of FedCSIS*. IEEE, Lodz, Poland, 1247–1255.

[43] Liang Ma, Ting He, Ananthram Swami, Don Towsley, and Kin K Leung. 2015. On optimal monitor placement for localizing node failures via network tomography. *Performance Evaluation* 91 (2015), 16–37.

[44] Liang Ma, Ting He, Ananthram Swami, Don Towsley, and Kin K Leung. 2017. Network capability in localizing node failures via end-to-end path measurements. *IEEE/ACM Trans. Netw.* 25, 1 (2017), 434–450.

[45] Liang Ma, Ting He, Ananthram Swami, Don Towsley, Kin K Leung, and Jessica Lowe. 2014. Node failure localization via network tomography. In *Proc. of IMC*. ACM, Vancouver,Canada, 195–208.

[46] Qiang Ma, Kebin Liu, Zhichao Cao, Tong Zhu, and Yunhao Liu. 2015. Link scanner: Faulty link detection for wireless sensor networks. *IEEE Trans. Wireless Commun.* 14, 8 (2015), 4428–4438.
[47] Arunanshu Mahapatro and Pabitra Mohan Khilar. 2013. Energy-efficient distributed approach for clustering-based fault detection and diagnosis in image sensor networks. *IET Wireless Sensor Systems* 3, 1 (2013), 26–36.
[48] Arunanshu Mahapatro and Pabitra Mohan Khilar. 2013. Fault diagnosis in wireless sensor networks: A survey. *IEEE Commun. Surveys Tuts.* 15, 4 (2013), 2000–2026.
[49] Ramona Marfievici, Amy L Murphy, Gian Pietro Picco, Federico Ossi, and Francesca Cagnacci. 2013. How environmental factors impact outdoor wireless sensor networks: a case study. In *Proc. MASS*. IEEE, 565–573.
[50] Anthéa Mayzaud, Rémi Badonnel, and Isabelle Chrisment. 2017. A distributed monitoring strategy for detecting version number attacks in RPL-based networks. *IEEE Trans. Netw. Service Manag.* 14, 2 (2017), 472–486.
[51] Amjad Mehmood, Nabil Alrajeh, Mithun Mukherjee, Salwani Abdullah, and Houbing Song. 2018. A survey on proactive, active and passive fault diagnosis protocols for wsns: network operation perspective. *Sensors* 18, 6 (2018), 1787.
[52] Xin Miao, Kebin Liu, Yuan He, Dimitris Papadias, Qiang Ma, and Yunhao Liu. 2013. Agnostic diagnosis: Discovering silent failures in wireless sensor networks. *IEEE Trans. Wireless Commun.* 12, 12 (2013), 6067–6075.
[53] Mobashir Mohammad, Manjunath Doddavenkatappa, and Mun Choon Chan. 2017. Improving performance of synchronous transmission-based protocols using capture effect over multichannels. *ACM Trans. Sen. Netw.* 13, 2 (2017), 1–26.
[54] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. 2010. Mining roles with multiple objectives. *ACM Trans. Inf. Syst. Secur.* 13, 4 (2010), 36.
[55] Thaha Muhammed and Riaz Ahmed Shaikh. 2017. An analysis of fault detection strategies in wireless sensor networks. *J. Netw. Comput. Appl.* 78 (2017), 267–287.
[56] Kapileswar Nellore and Gerhard P Hancke. 2016. A survey on urban traffic management system using wireless sensor networks. *Sensors* 16, 2 (2016), 157.
[57] Frances J Newbery. 1989. Edge concentration: A method for clustering directed graphs. *SIGSOFT Softw. Eng. Notes* 14, 7 (1989), 76–85.
[58] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. 2002. Random graph models of social networks. *Proc. Natl. Acad. Sci. U.S.A* 99, suppl 1 (2002), 2566–2572.
[59] Yan Qiao, Jun Jiao, Xinhong Cui, and Yuan Rao. 2020. Robust loss inference in the presence of noisy measurements and hidden fault diagnosis. *IEEE/ACM Trans. Netw.* 28, 1 (2020), 43–56.
[60] Fanrong Shi, Xianguo Tuo, Simon X Yang, Huailiang Li, and Rui Shi. 2017. Multiple two-way time message exchange (ttme) time synchronization for bridge monitoring wireless sensor networks. *Sensors* 17, 5 (2017), 1027.
[61] Andrey Somov, Alexander Baranov, and Denis Spirjakin. 2014. A wireless sensor–actuator system for hazardous gases detection and control. *Sens. Actuator A Phys.* 210 (2014), 157–164.
[62] Zhenyu Song, Longfei Shangguan, and Kyle Jamieson. 2017. Wi-fi goes to town: Rapid picocell switching for wireless transit networks. In *Proc. ACM SIGCOMM*. 322–334.
[63] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. 2004. Measuring ISP Topologies with Rocketfuel. *IEEE/ACM Trans. Netw.* 12, 1 (Feb. 2004), 2–16.
[64] Sava Stanic, Suresh Subramaniam, Gokhan Sahin, Hongsik Choi, and Hyeong-Ah Choi. 2010. Active monitoring and alarm management for fault localization in transparent all-optical networks. *IEEE Trans. Netw. Service Manag.* 7, 2 (2010), 118–131.
[65] Rakesh Ranjan Swain, Pabitra Mohan Khilar, and Sourav Kumar Bhoi. 2018. Heterogeneous fault diagnosis for wireless sensor networks. *Ad Hoc Netw.* 69 (2018), 15–37.
[66] Peng Tang and Tommy Chow. 2016. Wireless sensor-networks conditions monitoring and fault diagnosis using neighborhood hidden conditional random field. *IEEE Trans. Ind. Informat.* 12, 3 (2016), 933–940.
[67] Ivana Tomić, Po-Yu Chen, Michael J. Breza, and Julie A. McCann. 2018. Antilizer: Run Time Self-Healing Security for Wireless Sensor Networks. In *Proc. EAI MobiQuitous*. ACM, New York, 107–116.
[68] Ivana Tomić and Julie A McCann. 2017. A survey of potential security issues in existing wireless sensor network protocols. *IEEE Internet Things J.* 4, 6 (2017), 1910–1923.
[69] Satish Vadlamani, Burak Eksioglu, Hugh Medal, and Apurba Nandi. 2016. Jamming attacks on wireless networks: A taxonomic survey. *Int. J. Prod. Econ.* 172 (2016), 76–94.
[70] Hjalmar Wennerström, Frederik Hermans, Olof Rensfelt, Christian Rohner, and Lars-Åke Nordén. 2013. A long-term study of correlations between meteorological conditions and 802.15.4 link performance. In *Proc. of IEEE SECON*. IEEE, New Orleans, US, 221–229.
[71] Cheng-Bo Yu, Jing-Jing Hu, Rui Li, Shun-Hua Deng, and Ru-Min Yang. 2014. Node fault diagnosis in WSN based on RS and SVM. In *Proc. icWCSN*. IEEE, Wuhan, China, 153–156.
[72] M Yuvaraja and M Sabrigiriraj. 2017. Fault detection and recovery scheme for routing and lifetime enhancement in WSN. *Wireless Networks* 23, 1 (2017), 267–277.

[73] Ruoxi Zhang, Sara Newman, Marco Ortolani, and Simone Silvestri. 2018. A network tomography approach for traffic monitoring in smart cities. *IEEE Trans. Intell. Transp. Syst.* 19, 7 (2018), 2268–2278.
[74] Zeyu Zhang, Amjad Mehmood, Lei Shu, Zhiqiang Huo, Yu Zhang, and Mithun Mukherjee. 2018. A survey on fault diagnosis in wireless sensor networks. *IEEE Access* 6 (2018), 11349–11364.
[75] Shanshan Zheng and John S Baras. 2011. Trust-assisted anomaly detection and localization in wireless sensor networks. In *Proc. of SECON*. IEEE, Salt Lake City, US, 386–394.