On Optimal Monitor Placement for Localizing Node Failures via Network Tomography

Liang Ma[†], Ting He[†], Ananthram Swami[§], Don Towsley[*], and Kin K. Leung[‡]

[†]IBM T. J. Watson Research Center, Yorktown, NY, USA. Email: {maliang, the}@us.ibm.com

[§]Army Research Laboratory, Adelphi, MD, USA. Email: ananthram.swami.civ@mail.mil

[*]University of Massachusetts, Amherst, MA, USA. Email: towsley@cs.umass.edu

[‡]Imperial College, London, UK. Email: kin.leung@imperial.ac.uk

Abstract

We investigate the problem of placing monitors to localize node failures in a communication network from binary states (normal/failed) of end-to-end paths, under the assumption that a path is in normal state if and only if it contains no failed nodes. To uniquely localize failed nodes, the measurement paths must show different symptoms (path states) under different failure events. Our goal is to deploy the minimum set of monitors to satisfy this condition for a given probing mechanism. We consider three families of probing mechanisms, according to whether measurement paths are (i) arbitrarily controllable, (ii) controllable but cycle-free, or (iii) uncontrollable (i.e., determined by the default routing protocol). We first establish theoretical conditions that characterize network-wide failure identifiability through a per-node identifiability measure that can be efficiently evaluated for the above three probing mechanisms. Leveraging these results, we develop a generic monitor placement algorithm, applicable under any probing mechanism, that incrementally selects monitors to optimize the per-node measure. The proposed algorithm is shown to be optimal for probing mechanism (i), and provides upper and lower bounds on the minimum number of monitors required by the other probing mechanisms. In the special case of single-node failures, we develop an improved monitor placement algorithm that is optimal for probing mechanism (ii) and has linear time complexity. Using these algorithms, we study the impact of the probing mechanism on the number of monitors required for uniquely localizing node failures. Our results based on real network topologies show that although more complicated to implement, probing mechanisms that allow monitors to control measurement paths substantially reduce the required number of monitors.

I. Introduction

Effective monitoring of network performance is essential for network operators to build a reliable communication network robust against service disruptions. In order to achieve this goal, the monitoring infrastructure must be able to detect network misbehaviors (e.g., unusually high loss/latency, unreachability) and localize the sources (e.g., malfunction of certain routers) of these misbehaviors in an accurate and timely manner. Knowledge of where problematic

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

network elements reside in the network is particularly useful for fast service recovery, e.g., migration of affected services and/or rerouting of traffic. However, localizing network elements that cause a service disruption is challenging. The straightforward approach of directly monitoring the health of individual network elements incurs a high traffic overhead and is not always feasible due to access control or lack of protocol support at internal nodes. Moreover, built-in monitoring agents running on network elements cannot detect problems caused by unanticipated interactions between network layers, where end-to-end communication is disrupted but individual network elements along the path remain functioning (a.k.a. *silent failures*) [1]. These limitations call for a new approach to diagnose the health of network elements based on the health of end-to-end communications perceived between measurement points.

This different approach is generally known as *network tomography* [2], where a canonical application infers internal network characteristics by measuring end-to-end performance from a subset of nodes with monitoring capabilities, referred to as *monitors*. Unlike the approach of direct measurements that employ control packets, network tomography only relies on end-to-end performance (e.g., path connectivity) experienced by data packets, thus capable of reducing overhead, minimizing dependence on protocols, and detecting silent failures. In cases where the network characteristic of interest is binary (e.g., *normal* or *failed*), the problem is known as *Boolean network tomography* [3].

Given binary observations on paths (normal/failed), it is usually impossible to uniquely identify the states of individual network elements (nodes/links). For example, if two elements always appear together in any measurement path that contains one of them, then upon observing failures on these paths, we can at most infer that one (or both) of these elements has failed but not which one. Most existing works [1], [2], [4] address this uncertainty by focusing on a most probable solution that explains all path observations with the minimum number of failures. There is no guarantee, however, on how well this solution approximates the true failure locations.

Generally, to distinguish two possible sets of failures, there must exist a measurement path that traverses at least one element in one set and none of the elements in the other set. It is highly nontrivial to place monitors, such that this condition is satisfied with minimum cost, due to the large solution space (all combinations of monitor locations) and large number of constraints (all pairs of sets of failure locations). Several heuristics have been proposed to place monitors to uniquely localize a bounded number of link failures under specific probing mechanisms (e.g., traceroute) [5]–[7]. There is, however, a lack of understanding on the minimum number of monitors required for a generic probing mechanism and how this number varies for different probing mechanisms and different bounds on the number of failures.

In this paper, we apply Boolean network tomography to localize node failures. Node failures can be used to model failures of both physical nodes and links, with the latter represented as logical "nodes" connected to endpoints of the corresponding links. We consider the following problem: What is the minimum set of nodes that should be employed as monitors, such that *any* set of up to k node failures can be uniquely localized from the states of paths that are measurable under a given probing mechanism? We study this problem in the context of three families of probing mechanisms: (i) *Controllable Arbitrary-path Probing (CAP)*, where monitors can measure arbitrary paths subject to connectivity, (ii) *Controllable Simple-path Probing (CSP)*, where measurements are limited to cycle-free paths between monitors, and (iii) *Uncontrollable Probing (UP)*, where only paths between monitors selected by the default routing protocol can be measured. These probing mechanisms assume different levels of control over the routing of probes and are feasible in different network scenarios (see Section II-C). A comparison of the minimum numbers of monitors required to uniquely localize the same number of failures in the same topology under different

probing mechanisms thus establishes a fundamental tradeoff between the controllability of probing mechanism and the cost of monitor deployment.

A. Related Work

Existing works on tomography-based failure localization mostly consider link failures. Given path measurements, the standard approach is to find the minimum set of links whose failures explain all of the measurements, which gives the most probable set of failure locations under the assumption that failures are low-probability events. Using this approach, [2], [4] propose solutions for networks with tree topologies, which are later extended to general topologies by [1]. In a Bayesian formulation, [8] proposes a two-staged solution that first estimates the failure probabilities of different links and then infers the most likely set of failed links using subsequent measurements. Augmenting path measurements with available control plane information (e.g., routing messages), [9], [10] propose a greedy heuristic for troubleshooting network unreachability that has better accuracy than benchmarks using only path measurements.

Our work belongs to a complementary line of works that aim at *uniquely* localizing failures by strategically placing monitors. The optimal solution is known to be hard when monitors cannot control the routing of probes. Specifically, under round-trip probing (e.g., ping, traceroute) where only sources of probes need to be monitors, [5] shows that the optimal monitor placement is NP-hard and proposes a greedy approximation algorithm. Under the same probing mechanism, [6] shows that path selection can be solved in polynomial time, but monitor placement remains NP-hard. In the context of all-optical networks, [7] studies a similar problem of deactivating redundant monitors such that all failures remain uniquely identifiable by the remaining monitors, which again can be shown to be NP-hard. The probing mechanisms considered in these papers are most similar to UP, although UP requires both endpoints of measurement paths to be monitors (one-way probing) to avoid the inaccuracy of round-trip probing caused by route asymmetry. This introduces an additional challenge in monitor placement as paths measurable by one monitor under UP depend on locations of other monitors.

Interestingly, when measurement paths can be controlled (subject to certain constraints), the optimal monitor placement becomes computable in polynomial time. In the context of all-optical networks, where monitors can measure the states of arbitrary paths/cycles (a.k.a. "m-trail"), [11] proves that the network must be $(k+2)$-edge-connected to identify any failures of up to k links using one monitor, which is then used to develop a polynomial-time algorithm to place the minimum number of monitors for general topologies. When monitors can measure arbitrary paths with repeated links (a.k.a. "m-tour"), [12] shows that the requirement for identifying up to k link failures using one monitor is relaxed to k-edge connectivity, based on which a heuristic algorithm is proposed to place monitors for general topologies. The algorithm is then used to compare the numbers of monitors required under m-tour and m-trail probing. Our work shares a similar goal of [12] in that we are also interested in comparing monitor requirements under different probing mechanisms, but we study the problem in a different context of computer communication networks and for a different objective of localizing node failures. The first difference leads to different probing mechanisms, for which we adopt models proposed in [13] (CAP, CSP, and UP); see detailed discussions in Section II-C. The second difference leads to a critically different requirement that the solution should be robust to the failures of monitors themselves. Note that although one can represent each node failure by the failures of all its neighboring links and apply the results for link failure localization to localize node failures, the result will generally be suboptimal (i.e., placing more monitors than necessary), as we may be able to determine the state of a node without determining the states of *all* its neighboring links.

Necessary/sufficient conditions are established in [13] to uniquely localize a given number of node failures under CAP, CSP, and UP, respectively, which can be used to test whether a given monitor placement satisfies a desired level of identifiability. It is, however, highly nontrivial to compute the minimum monitor placement that satisfies these conditions, as there are exponentially many possible monitor placements to test. In this paper, we adopt the models in [13] to investigate efficient monitor placement algorithms under the above probing mechanisms.

A related but fundamentally different line of work is to study the minimum number of measurement paths needed to uniquely localize a given number of (node/link) metrics, using a CAP-like probing mechanism. Regarding this issue, [14] proposes a graph-constrained group testing solution, while [15] employs compressed sensing techniques to recover additive metrics (e.g., delay) in up to k nodes. In contrast, we only consider binary node/path states and seek to optimally deploy monitors such that node states can be uniquely localized using path measurements under a variety of probing mechanisms. A followup problem to monitor placement is how to select/construct measurement paths, for which various heuristic [5], [11], [12] and optimal [6] algorithms have been proposed. We leave the problem of constructing paths for localizing node failures under CAP/CSP/UP to future work.

B. Summary of Contributions

We study minimum monitor placement to uniquely localize *node* failures from binary end-to-end measurements under probing constraints. Our contributions are:

1) We establish a fine-grained measure of a network's capability to uniquely localize node failures as a function of network topology, monitor placement, and probing constraints.

2) Using the proposed measure, we develop a generic algorithm that incrementally places monitors such that all failures up to a given bound can be uniquely identified using an arbitrary probing mechanism. We then provide polynomial-time implementations of this algorithm for specific probing mechanisms (CAP, CSP, and UP). We prove that the proposed algorithm is optimal under CAP, and provides upper and lower bounds on the minimum number of monitors under CSP and UP.

3) For CSP, we further develop a linear-time algorithm that selects the minimum number of monitors to uniquely identify any single-node failure.

4) We compare the monitor requirements under different probing mechanisms on real topologies. Our study shows that although controllable probing, especially CAP, is more difficult to implement, it significantly reduces the number of monitors compared to more restrictive probing mechanisms.

Note that we have limited our observations to binary states (normal/failed) of measurement paths. It is possible in some networks to obtain extra information, e.g., rerouted paths or correlation of failures, in which case our solution provides upper bounds on the minimum monitor requirement.

The rest of the paper is organized as follows. Section II formulates the problem. Section III proposes a fine-grained measure of identifiability, based on which Section IV presents an efficient monitor placement algorithm that is provably optimal under CAP and provides bounds for other probing mechanisms. In Section V, we further investigate the case of single-node failures and develop an optimal monitor placement algorithm for CSP. All proposed algorithms are evaluated on real topologies in Section VI to compare monitor requirements under different probing mechanisms. Finally, Section VII concludes the paper.

II. Problem Formulation

A. Models and Assumptions

We assume that the network topology is known and can be modeled as an undirected graph[1] $\mathcal{G} = (V, L)$, where V and L are the sets of nodes and links. In $\mathcal{G}$, the number of neighbors of node v is called the degree of v. Without loss of generality, we assume that $\mathcal{G}$ is connected, as different connected components have to be monitored separately.

A subset of nodes $M \subseteq V$ (determined by monitor placement) are *monitors* that can initiate and collect measurements. The rest of the nodes, denoted by $N := V \setminus M$, are *non-monitors*. All nodes (monitors and non-monitors) in V may fail unexpectedly, and a failure event may involve multiple nodes. We use *node state* (*path state*) to refer to the state, failed or normal, of nodes (paths), where a path fails if and only if at least one node on the path fails. We assume that monitors in normal states report measured path states to a monitoring center via uninterrupted control plane messaging (even if the paths in data plane may be disrupted by failures)[2]. Consequently, the states of monitors can always be uniquely determined. Depending on the adopted probing mechanism, we can measure and infer the states of non-monitors by sending probes along selected measurement paths using the normal monitors. Let P denote the set of all *possible measurement paths* under a given probing mechanism. This includes both paths for sending probes between monitors and degenerate paths corresponding to individual monitors (since monitors can be measured directly), i.e., $P = \{m : m \in M\} \cup \{\mathcal{P}_{m_i m_j} : m_i \in M, m_j \in M\}$, where $\mathcal{P}_{m_i m_j}$ is a measurement path between monitors m_i and m_j allowed by the probing mechanism. Given $\mathcal{G}$ and M, different probing mechanisms can lead to different measurement paths, which will be discussed in Section II-C.

B. Definitions

Let a *failure set* be a set of nodes (monitors or non-monitors) that can fail simultaneously. Two failure sets may overlap. The goal of failure localization is therefore to determine the failure set from the states of measurement paths. The challenge is that there may exist multiple failure sets leading to the same path states, causing an ambiguity. Let $P_F \subseteq P$ denote the set of all measurement paths affected by a given failure set F, i.e., traversing at least one node in F. We borrow the following definition from [13] to formalize the requirement for unique failure localization.

Definition 1: [13] Given a network $\mathcal{G}$, a set of measurement paths P, and a collection Ψ of potential failure sets in $\mathcal{G}$:

1) Two failure sets F_1 and F_2 are *distinguishable* using monitors in M if and only if $P_{F_1} \neq P_{F_2}$, i.e., $\exists$ a measurement path that traverses one and only one of F_1 and F_2.
2) In Ψ, failure set F is *identifiable* if and only if F is distinguishable from every other failure set in Ψ.
3) Ψ is *identifiable* if and only if every failure set in Ψ is identifiable.

It is clear from Definition 1 that whether a failure set is identifiable or not depends on the collection of potential failure sets it is compared against. Under the assumption that failures are low-probability events, one can define Ψ to include probable failure sets by bounding the cardinality of failure sets. Formally, we adopt the following definition from [13].

Definition 2: [13] Given a network $\mathcal{G}$ and a set of measurement paths P in $\mathcal{G}$, we say that $\mathcal{G}$ is k-*identifiable* ($0 \leq k \leq |V|$) if the collection Ψ of all node sets of size up to k is identifiable, i.e., any failure of up to k nodes (monitors or non-monitors) can be uniquely localized.

[1]We use the terms *network* and *graph* interchangeably.

[2]This is feasible for soft failures caused by congestion and busy CPU, as each report only contains a few bits of information.

The notion of k-identifiability is a worst-case guarantee. That is, as long as the total number of failures is bounded by k, we can uniquely localize the failures from observed path states in a k-identifiable network no matter where the failures occur. Note that there is a subtle but critical difference between our formulation and the formulation in [13] in that [13] assumes monitors do not fail. Since failures of monitors can always be localized, an equivalent definition of k-identifiability is that under any set of failed monitors $F_M \subseteq M$ with $|F_M| \leq k$, failures of up to $k - |F_M|$ non-monitors can be uniquely localized using path measurements from the remaining monitors.

C. Classification of Probing Mechanisms

Given topology $\mathcal{G}$ and the monitor locations M, the probing mechanism plays a crucial role in determining the set of measurement paths P. To study the impact of probing mechanisms on monitor placement, we consider the following three families of probing mechanisms defined in [13]:

1) *Controllable Arbitrary-path Probing (CAP):* P includes any path/cycle, allowing repeated nodes/links, as long as each path/cycle starts and ends at monitors.
2) *Controllable Simple-path Probing (CSP):* P includes any degenerate path of a single monitor, and any *simple* (i.e., cycle-free) path between distinct monitors.
3) *Uncontrollable Probing (UP):* P includes each degenerate single-monitor path and a path between each pair of monitors specified by the routing protocol for data flows.

These probing mechanisms provide different controllability of probes and are feasible in different network environments. For example, CAP is feasible under *source routing* or in *software-defined networks (SDN)*, CSP is feasible under MPLS (MultiProtocol Label Switching) routing, and UP is feasible in any communication network employing stable single-path routing; see detailed discussions in [13].

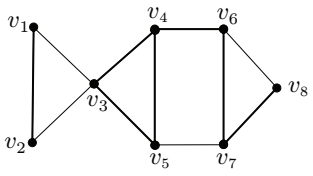
Discussion: Although CAP allows probes to traverse each link an arbitrary number of times, it suffices to consider paths that traverse each link at most once in each direction for the sake of localizing failures, and thus CAP is equivalent to "m-tour" probing in [12]. In all-optical networks, [11] considers another probing mechanism called "m-trail", where measurement paths can contain repeated nodes but *not* repeated links. It is unclear what routing protocols in general communication networks select paths according to "m-trails"; we therefore do not consider such probing mechanism in this paper.

D. Objective

Given a network topology $\mathcal{G}$, a probing mechanism (CAP, CSP, or UP), and a maximum number of failures k, we want to select the minimum set of nodes in $\mathcal{G}$ as monitors (i.e., selecting $M \subseteq V$) such that $\mathcal{G}$ is k-identifiable. Here k is an input parameter capturing the scale of failures the system is designed to handle. Clearly, a more restrictive probing mechanism (UP vs. CSP, CSP vs. CAP) requires more monitors to be placed. It is, however, non-trivial to quantify the gap in the required number of monitors and how it depends on other parameters including the maximum number of failures and the network topology. Our goal is to quantify this gap by developing and evaluating efficient monitor placement algorithms.

E. Illustrative Example

Consider the problems of placing monitors in the sample network in Fig. 1 to achieve 1-identifiability and 2-identifiability ($k = 1, 2$), respectively. Under UP, suppose that the default routing protocol allows the following paths:



Default routing paths under UP
$\mathcal{P}_1$=$v_1v_3v_4v_6v_8$
$\mathcal{P}_2$=$v_2v_3v_5v_7v_8$
$\mathcal{P}_3$=$v_1v_3v_4v_5$
$\mathcal{P}_4$=$v_1v_3v_2$
$\mathcal{P}_5$=$v_2v_3v_5$
$\mathcal{P}_6$=$v_5v_4v_6$

Fig. 1. Monitor placement in a sample network. (i) For 1-identifiability: $M = \{v_1\}$ under CAP, $M = \{v_1, v_8\}$ under CSP, and $M = \{v_1, v_2, v_5, v_8\}$ under UP. (ii) For 2-identifiability: $M = \{v_1, v_4\}$ under CAP, $M = \{v_1, v_2, v_4, v_8\}$ under CSP, and $M = \{v_1, v_2, v_5, v_6, v_8\}$ under UP.

$\mathcal{P}_1 = v_1v_3v_4v_6v_8$, $\mathcal{P}_2 = v_2v_3v_5v_7v_8$, $\mathcal{P}_3 = v_1v_3v_4v_5$, $\mathcal{P}_4 = v_1v_3v_2$, $\mathcal{P}_5 = v_2v_3v_5$, and $\mathcal{P}_6 = v_5v_4v_6$. By selecting $M = \{v_1, v_2, v_5, v_8\}$ as monitors, we can verify that the network is 1-identifiable (i.e., for every two nodes, there is a measurement path traversing one and only one of them), and no placement can achieve 1-identifiability with fewer monitors. For 2-identifiability, additional monitors must be selected, e.g., selecting v_6 as an additional monitor achieves 2-identifiability. The required number of monitors can be reduced under a more flexible probing mechanism. For example, under CSP, selecting $M = \{v_1, v_8\}$ as monitors suffices to achieve 1-identifiability, saving two monitors compared with UP. To achieve 2-identifiability under CSP, we can select v_2 and v_4 as additional monitors, again saving a monitor compared with UP. If CAP is supported, then we can further reduce the number of monitors: selecting v_1 (or any other node) as the only monitor, we can localize any single-node failure by constructing nested measurement paths starting and terminating at v_1, whereby 1-identifiability is achieved; it can also be verified that only two monitors, e.g., $M = \{v_1, v_4\}$, are sufficient to achieve 2-identifiability.

This example shows that in addition to the network topology and the maximum number of failures, the probing mechanism also significantly affects the required number of monitors. In the rest of the paper, we will study this impact both theoretically and algorithmically.

III. Objective of Monitor Placement

Conceptually, our monitor placement algorithms try to solve the optimization problem of placing the minimum number of monitors subject to the constraint of satisfying k-identifiability. The definition of k-identifiability in Definition 2 is based on the enumeration of all possible failure scenarios and does not directly allow efficient testing. To address this issue, [13] establishes explicit conditions for k-identifiability that can be verified efficiently (in polynomial time) for each of the three probing mechanisms (CAP, CSP, UP). This result, however, still does not allow efficient optimization of monitor placement as there are exponentially many possible placements to test. We address this challenge by proposing a fine-grained measure of identifiability that can differentiate intermediate monitor placements and is thus suitable for an incremental algorithm that iteratively selects monitors to achieve k-identifiability.

A. Fine-grained Measure of Identifiability

We start by noticing that even if a network is not k-identifiable under a given monitor placement, we may still be able to uniquely determine the states of some nodes (e.g., monitors and immediate neighbors of normal monitors) no matter where the failures occur. This observation motivates us to establish fine-grained measures that can characterize the extent to which node states can be identified under failures. To this end, we adapt the (network-wide) k-identifiability definition in Definition 2 to the following per-node property.

Definition 3: Given a network $\mathcal{G}$ and a set of measurement paths P:

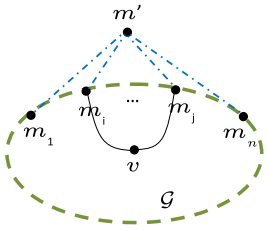


Fig. 2. Extended graph $\mathcal{G}'$, where m_i's are real monitors and m' is a virtual monitor.

1) Node v is *k-identifiable* if for any two failure sets F_1 and F_2 satisfying (1) $|F_i| \leq k$ $(i = 1, 2)$, (2) $F_1 \cap \{v\} \neq F_2 \cap \{v\}$, F_1 and F_2 are distinguishable.
2) The *maximum identifiability of* v, denoted by $\Omega(v)$, is the maximum value of k such that v is k-identifiable.

If v is a monitor, the degenerate path (v) can always distinguish between F_1 and F_2 for $v \in F_1$ and $v \notin F_2$; hence we define $\Omega(v) := |V|$ for monitors. In other words, the maximum identifiability of node v is the maximum number of failures that may happen anywhere in the network such that the state of node v can be uniquely determined from states of measurement paths. One should not confuse $\Omega(v)$ with the overall maximum identifiability of a network $\mathcal{G}$, denoted by $\Omega(\mathcal{G})$, which represents the maximum number of failures anywhere in the network such that the states of all nodes can be uniquely determined [13]. We will refer to $\Omega(v)$ as the *node* maximum identifiability and $\Omega(\mathcal{G})$ as the *network* maximum identifiability. The significance of maximum node identifiability is that we can express the k-identifiability of a network (Definition 2) in terms of conditions on the maximum identifiability of nodes.

Theorem 4: A network $\mathcal{G}$ is k-identifiable *if and only if* $\Omega(v) \geq k$ for each $v \in V$.

Proof: Necessity. Suppose $\exists$ node $v \in V$ that is not k-identifiable (i.e., $\Omega(v) < k$), then $\exists$ at least two failure sets F_1 and F_2 with $|F_i| \leq k$ $(i = \{1, 2\})$ and $F_1 \cap \{v\} \neq F_2 \cap \{v\}$ such that F_1 and F_2 are not distinguishable. By Definition 2, $\mathcal{G}$ is not k-identifiable.

Sufficiency. For any two failure sets F_1 and F_2 with $F_1 \neq F_2$ and $|F_i| \leq k$ $(i = \{1, 2\})$, $\exists$ a node v that is in one and only one of F_1 and F_2. Since node v is k-identifiable ($\Omega(v) \geq k$), F_1 and F_2 must be distinguishable by Definition 3. Therefore, $\mathcal{G}$ is k-identifiable. ■

Theorem 4 implies that an equivalent formulation for monitor placement is to select the minimum number of monitors such that each node has a maximum identifiability of at least k; we will use this observation to develop an efficient monitor placement algorithm (see Section IV-A). Note that by this theorem, we can relate node and network maximum identifiabilities by $\Omega(\mathcal{G}) = \min_{v \in V} \Omega(v)$.

Definition 3 still relies on the enumeration of all possible failure sets and therefore cannot be used to evaluate $\Omega(v)$ efficiently. In the rest of this section, we explore how to efficiently compute $\Omega(v)$ for each of the three probing mechanisms considered in this paper.

B. Maximum Node Identifiability under CAP

Our idea for computing $\Omega(v)$ under CAP, denoted by $\Omega^{\text{CAP}}(v)$, is based on the following observation. Consider an *extended graph* $\mathcal{G}'$ constructed by adding a virtual monitor m' that is connected to each real monitor via a virtual link; see Fig. 2. For a non-monitor v, let $C_{\mathcal{G}'}(v, m')$ denote the *minimum vertex cut* between v and m', i.e., the minimum node set that separates v and m' in $\mathcal{G}'$; if v is a monitor, define $C_{\mathcal{G}'}(v, m') := V$. Then we must have $\Omega(v) \leq |C_{\mathcal{G}'}(v, m')|$ because node v cannot be probed if all nodes in $C_{\mathcal{G}'}(v, m')$ fail. Below we show a stronger result that the bound is actually tight and characterizes the exact value of $\Omega^{\text{CAP}}(v)$.

Theorem 5: The maximum identifiability of node v under CAP is $\Omega^{\text{CAP}}(v) = |C_{\mathcal{G}'}(v, m')|$.

Proof: It suffices to show that non-monitor v is $|C_{\mathcal{G}'}(v,m')|$-identifiable. Consider any two failure sets F_1 and F_2 with $|F_i| \leq |C_{\mathcal{G}'}(v,m')|$ $(i=1,\ 2)$, $v \in F_1$, and $v \notin F_2$. Let $I := F_1 \cap F_2$. Since $|I| \leq |C_{\mathcal{G}'}(v,m')|-1$, $\exists$ a path $\mathcal{P}$ connecting v to m' in $\mathcal{G}'-I$. Let m be the first real monitor on $\mathcal{P}$ (starting from m') and w be the first node on $\mathcal{P}$ that is in either $F_1 \setminus I$ or $F_2 \setminus I$. Let $\mathcal{P}'$ denote the segment of $\mathcal{P}$ between m and w. Then $\mathcal{P}'$ and its reverse path form a measurement path from m to w and back to m that only traverses nodes in one of F_1 and F_2, thus distinguishing between F_1 and F_2. ■

Complexity: The problem of finding vertex-cut between v and m' in an undirected graph $\mathcal{G}'$, i.e., $C_{\mathcal{G}'}(v,m')$, can be reduced to an edge-cut problem between v and m' in a directed graph in linear time [16]. The v-to-m' edge-cut problem is solvable by the Ford−Fulkerson algorithm [17] in $O(|L|\cdot|M|)$ time. Therefore, the complexity for computing $\Omega^{\text{CAP}}(v)$ is $O(|L|\cdot|M|)$.

C. Maximum Node Identifiability under CSP

Under CSP, measurement paths have to start/end at different monitors and not contain repeated nodes. Nevertheless, we can apply similar observations as above to derive tight upper/lower bounds on $\Omega^{\text{CSP}}(v)$ as follows.

Theorem 6: The maximum identifiability of node v under CSP satisfies

a) $|C_{\mathcal{G}'}(v,m')|-2 \leq \Omega^{\text{CSP}}(v) \leq |C_{\mathcal{G}'}(v,m')|-1$, if $2 \leq |C_{\mathcal{G}'}(v,m')| < |V|$,

b) $\Omega^{\text{CSP}}(v)=0$, if $|C_{\mathcal{G}'}(v,m')| \leq 1$, and

c) $\Omega^{\text{CSP}}(v)=|V|$, if $|C_{\mathcal{G}'}(v,m')|=|V|$.

Proof: For any two failure sets F_1 and F_2 with $|F_i| \leq k$ $(i=\{1,2\})$, $v \in F_1$, and $v \notin F_2$, if $|C_{\mathcal{G}'}(v,m')| \geq k+2$, then $\exists$ vertex disjoint (except at v) paths from v to monitors in $\mathcal{G}-F_2$; concatenating these two paths generates a measurement path that can distinguish between F_1 and F_2. However, if $|C_{\mathcal{G}'}(v,m')| \leq k$, then $\exists$ a node w and a node set F' in $\mathcal{G}$ with $|F'| \leq k-1$, such that the removal of F' results in all paths from v to monitors going through w. Therefore, $\nexists$ cycle-free path that can distinguish between failure sets F' and $F' \cup \{v\}$ ($|F' \cup \{v\}| \leq k$). Thus, v is k-identifiable under CSP (i) if $|C_{\mathcal{G}'}(v,m')| \geq k+2$, and (ii) only if $|C_{\mathcal{G}'}(v,m')| \geq k+1$. Since $k=|C_{\mathcal{G}'}(v,m')|-2$ is the largest number that satisfies sufficient condition (i), and $k=|C_{\mathcal{G}'}(v,m')|$ is the smallest number that violates necessary condition (ii), we have $|C_{\mathcal{G}'}(v,m')|-2 \leq \Omega^{\text{CSP}}(v) \leq |C_{\mathcal{G}'}(v,m')|-1$ for $2 \leq |C_{\mathcal{G}'}(v,m')| < |V|$. Finally, if $|C_{\mathcal{G}'}(v,m')| \leq 1$, then $\nexists$ cycle-free path traversing v, and thus $\Omega^{\text{CSP}}(v)=0$; $\Omega^{\text{CSP}}(v)=|V|$ if v is a monitor, i.e., $|C_{\mathcal{G}'}(v,m')|=|V|$, according to Definition 3. ■

Theorem 6 states that although the exact value of $\Omega^{\text{CSP}}(v)$ cannot be directly determined in most cases, we can obtain upper and lower bounds that only differ by 1, thus limiting $\Omega^{\text{CSP}}(v)$ to one of two consecutive integers.

Complexity: Similar to CAP, computing $C_{\mathcal{G}'}(v,m')$ takes $O(|L|\cdot|M|)$ time. Therefore, the complexity for determining the lower and upper bounds for $\Omega^{\text{CSP}}(v)$ using Theorem 6 is $O(|L|\cdot|M|)$.

D. Maximum Node Identifiability under UP

Similar to CSP, we can bound the maximum identifiability $\Omega(v)$ under UP, denoted by $\Omega^{\text{UP}}(v)$, from both sides. Let $P_v \subseteq P$ denote the set of measurement paths traversing a node v, and $\mathcal{S}_v := \{P_w :\ w \in V,\ w \neq v\}$ denote the collection of path sets traversing nodes in $V \setminus \{v\}$. We use $\text{MSC}(v)$ to denote the cardinality of the *minimum set cover* of P_v by $\mathcal{S}_v$, i.e., $\text{MSC}(v) := |V'|$ for the minimum set $V' \subseteq V \setminus \{v\}$ such that $P_v \subseteq \bigcup_{w \in V'} P_w$. Note that this definition excludes the case that v itself is a monitor; if v is a monitor, then we define $\text{MSC}(v) := |V|$.

Intuitively, MSC(v) is the minimum number of failures at nodes other than v that will hide the failure of v (by disrupting all paths traversing v), and thus determines the value of $\Omega^{\text{UP}}(v)$.

Theorem 7: The maximum identifiability of node v under UP satisfies

a) MSC$(v) - 1 \leq \Omega^{\text{UP}}(v) \leq$ MSC(v) if $1 \leq$ MSC$(v) < |V|$,

b) $\Omega^{\text{CSP}}(v) = 0$ if MSC$(v) = 0$, and

c) $\Omega^{\text{CSP}}(v) = |V|$ if MSC$(v) = |V|$.

Proof: For any two failure sets F_1 and F_2 with $|F_i| \leq k$ $(i = \{1, 2\})$, $v \in F_1$, and $v \notin F_2$, if MSC$(v) \geq k+1$, then $\exists$ a measurement path $\mathcal{P} \in P_v \setminus \bigcup_{w \in F_2} P_w$, i.e., $\mathcal{P}$ goes through v but none of the nodes in F_2. Thus, $\mathcal{P}$ distinguishes F_1 and F_2. However, if MSC$(v) \leq k-1$, then $\exists$ a node set F' with $|F'| \leq k-1$, such that the removal of F' disconnects all paths traversing v under UP, and thus F' and $F' \cup \{v\}$ are not distinguishable. Thus, v is k-identifiable under UP (i) if MSC$(v) \geq k+1$, and (ii) only if MSC$(v) \geq k$. Since $k =$ MSC$(v) - 1$ is the largest number that satisfies sufficient condition (i), and $k =$ MSC$(v) + 1$ is the smallest number that violates necessary condition (ii), we have MSC$(v) - 1 \leq \Omega^{\text{UP}}(v) \leq$ MSC(v) when $1 \leq$ MSC$(v) < |V|$. Finally, if v is a monitor, i.e., MSC$(v) = |V|$, then $\Omega^{\text{UP}}(v) = |V|$ according to Definition 3; if v is not traversed by any measurement path, i.e., MSC$(v) = 0$, then obviously $\Omega^{\text{UP}}(v) = 0$. ■

Complexity: Evaluating $\Omega^{\text{UP}}(v)$ by Theorem 7 requires solving the *set covering problem*, which is NP-hard [18]. Nevertheless, we can use approximation algorithms to compute bounds on MSC(v). An algorithm with the best approximation guarantee is the *greedy algorithm*, which iteratively selects the set in $\mathcal{S}_v$ that contains the largest number of uncovered paths in P_v until all the paths in P_v are covered (for a non-monitor v). Let GSC(v) denote the number of sets selected by the greedy algorithm. This immediately provides an upper bound: MSC$(v) \leq$ GSC(v). Moreover, since the greedy algorithm has an approximation ratio of $\log(|P_v|) + 1$ [19], we can also bound MSC(v) from below: MSC$(v) \geq$ GSC$(v)/(\log(|P_v|) + 1)$. Computation of GSC(v) takes polynomial time: $O(|P_v|^2|V|) = O(|P|^2|V|)$ (or $O(|M|^4|V|)$ if P contains paths between all pairs of monitors).

IV. Monitor Placement for Multi-Failure Localization

By Theorem 4, an alternate monitor placement formulation with respect to (w.r.t.) a given maximum number of failures k is to ensure sufficiently large ($\geq k$) maximum identifiability $\Omega(v)$ for each node v by selecting a minimum set of nodes as monitors. This alternate formulation allows one to evaluate intermediate solutions by defining an objective function $f((\Omega(v))_{v \in V}|M)$ that maps $(\Omega(v))_{v \in V}$ under monitor placement M to a scalar, such that a monitor placement M_1 can be considered preferable to another placement M_2 if $f((\Omega(v))_{v \in V}|M_1) \geq f((\Omega(v))_{v \in V}|M_2)$ (even if neither placement achieves k-identifiability). Based on this idea, we develop an efficient monitor placement algorithm that incrementally selects monitors to optimize a particular objective function. We prove that the proposed algorithm is optimal for CAP and provides upper and lower bounds on the minimum number of monitors for CSP and UP.

A. Monitor Placement Based on Maximum Node Identifiability

The basic idea of the proposed monitor placement algorithm, called *Maximum Node-identifiability Monitor Placement (MNMP)*, is to select monitors iteratively, such that each selection maximizes the sum of $\min(\Omega(v), k)$ over $v \in V$; see Algorithm 1. MNMP is applicable to any probing mechanism. More importantly, MNMP is optimal under a specific family of probing mechanisms (see Theorem 9). MNMP has three steps:

Algorithm 1: Maximum Node-identifiability Monitor Placement (MNMP)

input : Network topology $\mathcal{G}$, parameter k ($1 \leq k \leq |V|$)

output: Set of monitors $M \subseteq V$

1 $M \leftarrow \emptyset$; // $\leftarrow$: assignment

2 $U \leftarrow V$; // U: uncovered nodes

3 **while** $U \neq \emptyset$ **do**

4 $\quad m = \arg\max_{w \in V \setminus M} |U \cap \mathcal{V}(w,\ M)|$;

5 $\quad U \leftarrow U \setminus \mathcal{V}(w,\ M)$;

6 $\quad M \leftarrow M \cup \{m\}$;

7 **end**

8 **while** $\exists v \in V$ *with* $\Omega(v) < k$ **do**

9 $\quad m^* \leftarrow \arg\max_{w \in V \setminus M} \sum_{v \in V} \min\big(\Omega(v|M \cup \{w\}), k\big)$;

10 $\quad M \leftarrow M \cup \{m^*\}$;

11 **end**

12 **for** *each* $m \in M$ **do**

13 $\quad M \leftarrow M \setminus \{m\}$ if $\Omega(v|M \setminus \{m\}) \geq k$ for each $v \in V$;

14 **end**

Step (1) First, MNMP selects monitors to ensure that all non-monitors can be covered by measurement paths. Given $M \subseteq V$ and $w \in V \setminus M$, define $\mathcal{V}(w,\ M)$ as the set of nodes covered by measurement paths starting from w, when selecting w as a monitor on top of existing monitors M, i.e., $\mathcal{V}(w,\ M)$ are all nodes on measurement paths between w and each $m \in M$. Based on this definition, lines 3–7 iteratively select a new monitor whose paths to the existing monitors cover the maximum number of uncovered nodes, until all nodes are covered by at least one measurement path.

Step (2) Next, MNMP selects additional monitors if needed so that $\Omega(v) \geq k$ for each $v \in V$. Let $\Omega(v|M)$ denote the maximum identifiability of node v under monitor placement M. Given a subroutine to evaluate $\Omega(v|M)$ (detailed later), lines 8–11 select new monitors iteratively such that each selection maximizes the sum maximum node identifiability, capped by k, i.e., $\sum_{v \in V} \min\big(\Omega(v|M \cup \{w\}), k\big)$. The iteration continues until $\Omega(v) \geq k \forall v \in V$, i.e., $\mathcal{G}$ is k-identifiable.

Step (3) Finally, MNMP goes through selected monitors in an arbitrary order to check if a monitor can be removed from M without violating k-identifiability, and removes it if so (lines 12–14).

In MNMP, the time complexity for computing $\mathcal{V}(w, M)$ (line 4) and $\Omega(v|M)$ (lines 9, 13) varies for different probing mechanisms. In the following section, we discuss how to implement MNMP efficiently under three specific probing mechanisms (CAP, CSP, and UP), and examine the associated properties of MNMP.

1) Implementation under CAP: Since the given network $\mathcal{G}$ is a connected graph, for any non-monitor v and monitor m in $\mathcal{G}$, there exists a path $\mathcal{P}$ connecting v and m; moreover, CAP allows each link to be traversed multiple times by the same probe, and thus a probe starting from m can be sent to v along path $\mathcal{P}$ and returned to m via the same path, i.e., a single monitor can probe any non-monitor. Therefore, selecting any node as monitor ensures that the uncovered node set U in line 3 is empty (i.e., complexity of lines 3–7 is reduced to $O(1)$). When selecting additional monitors (lines 8–11) or deselecting redundant monitors (lines 12–14), MNMP-CAP uses a subroutine that computes the size of minimum vertex cut to evaluate $\Omega(v|M)$ in polynomial time based on the result in Theorem 5.

Complexity: Section III-B shows the complexity for computing $\Omega^{\text{CAP}}(v)$ is $O(|L| \cdot |M|)$. Thus, lines 8–11 take $O(|L| \cdot |V| \cdot |M|^2)$ time, and lines 12–14 take $O(|L| \cdot |M| \cdot |V| \cdot |M|)$ time. The overall complexity of MNMP-CAP

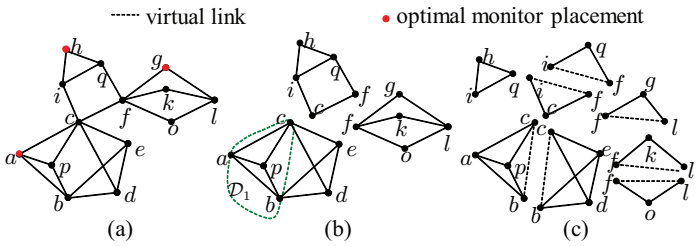


Fig. 3. Graph decomposition: (a) Sample network; (b) Biconnected components in (a); (c) Triconnected components in (a).

is thus $O(|L| \cdot |V| \cdot |M|^2)$, or $O(|L| \cdot |V|^3)$ in the worst case.

2) Implementation under CSP: Unlike CAP, a single monitor cannot probe any node (except for the monitor itself) under CSP. Therefore, multiple monitors are needed to ensure coverage (lines 3–7). Because there can be exponentially many paths between each pair of nodes under CSP, we cannot compute $\mathcal{V}(w, M)$ by simply enumerating paths between a candidate monitor w and existing monitors M. To efficiently compute $\mathcal{V}(w, M)$, we need to first understand how the network topology affects the coverage of paths. We introduce the following definition.

Definition 8: [20] (a) Graph $\mathcal{G}$ containing n vertices is k-connected if $k \leq n-1$ and deleting any subset of up to $k-1$ vertices does not disconnect $\mathcal{G}$.

(b) A *q-connected component* of $\mathcal{G}$ is a maximal subgraph of $\mathcal{G}$ that is either q-connected, or a complete graph (i.e., a clique) with up to q vertices. A 2-connected (3-connected) component is also called a biconnected (triconnected) component.

Based on Definition 8, a biconnected component is either a 2-connected subgraph or a *bond* (i.e., two nodes connected by a single link). All biconnected components are connected by *cut-vertices*, each being a vertex whose removal disconnects the graph. For instance, Fig. 3 (b) shows the biconnected components of Fig. 3 (a), separated by cut-vertices c and f.

Based on this concept, it is clear that each biconnected component with fewer than two cut-vertices must have at least one monitor; otherwise, non-cut-vertices in this component cannot be measured using cycle-free paths. Meanwhile, if each biconnected component has at least two nodes that are cut-vertices or monitors, then it can be verified that every non-monitor is traversed by a simple path between monitors. Motivated by this observation, lines 3–7 can be implemented as follows: compute biconnected components of $\mathcal{G}$ [21], and randomly select a non-cut-vertex node as a monitor in each biconnected component with fewer than two cut-vertices; if $\mathcal{G}$ contains only one biconnected component (i.e., $\mathcal{G}$ is 2-connected), randomly select two nodes as monitors. The selected monitors ensure that all nodes are covered by measurement paths allowed by CSP (degenerate paths or simple paths between monitors).

To implement lines 9 and 13, we resort to the result in Theorem 6. Since we only have lower/upper bounds for $\Omega^{\text{CSP}}(v)$, we use the lower bound in evaluating the value of $\Omega^{\text{CSP}}(v|M)$ so that the constructed monitor set M is always sufficient for achieving k-identifiability under CSP.

Complexity: Lines 3–7 takes $O(|V|)$ time, dominated by the time to find biconnected components [21]. Computing the lower bound of $\Omega^{\text{CSP}}(v)$ takes $O(|L| \cdot |M|)$ (shown in Section III-C). Thus, similar to CAP, the overall complexity for MNMP-CSP is $O(|L| \cdot |V| \cdot |M|^2) = O(|L| \cdot |V|^3)$.

3) Implementation under UP: To place monitors under UP, we need to know, besides the network topology $\mathcal{G}$, the set of *routable paths* $Q := (q_{uv})_{u,v \in V}$ between all pairs of nodes (e.g., by knowing the routing protocol), where q_{uv} is a routable path between nodes u and v.

To cover all nodes by measurement paths under UP, we observe that: (i) all degree-one nodes must be monitors, and (ii) at least one in every two consecutive degree-two nodes must be a monitor (assuming routable paths are cycle-free). This observation allows us to bootstrap MNMP-UP by initializing M with nodes satisfying the above conditions. If there are still uncovered nodes, then we choose additional monitors according to lines 3–7, where $\mathcal{V}(w,\ M) := \bigcup_{m\in M} q_{wm}$ (viewing q_{wm} as the set of nodes on path q_{wm}).

We apply the result of Theorem 7 to evaluate $\Omega^{\text{UP}}(v|M)$. Again, since we only have lower/upper bounds, we use the lower bound to represent the value of $\Omega^{\text{UP}}(v|M)$. The exact lower bound is NP-hard to evaluate, and a greedy heuristic can be applied to evaluate a relaxed lower bound as discussed in Section III-D. Although theoretically the relaxed bound differs from the true value of $\Omega^{\text{UP}}(v|M)$ by a logarithmic factor (i.e., $1/(\log|P_v|+1)$), empirical studies in [13] have shown that the greedy heuristic gives near-optimal results that can be used to approximate the original, tight lower bound (differing from $\Omega^{\text{UP}}(v|M)$ by at most one).

Complexity: Lines 3–7 contain $O(|M|)$ iterations, each taking $O(|M|\cdot|V|^2)$ time dominated by line 4; lines 8–11 and lines 12–14 each have $O(|M|)$ iterations, each iteration taking $O(|P|^2\cdot|V|^2)$ time (using alternate greedy heuristic) dominated by lines 9 and 13. Therefore, the overall complexity of MNMP-UP is $O(|M|\cdot|P|^2\cdot|V|^2) = O(|P|^2\cdot|V|^3)$.

Discussion: A similar problem of Redundant Monitor Deactivation Problem (RMDP) is investigated in [7] to remove redundant monitors such that remaining monitors maintain the same failure localization capability, which is shown to be NP-complete and solved by a greedy heuristic. RMDP is based on a different observation model where monitors observe states of paths traversing them, and thus paths measured by a monitor are independent of other monitors. In contrast, monitors in our problem measure states of paths starting/ending at them, and thus paths measured by a monitor depend on which other nodes are monitors. Therefore, the solution to RMDP cannot be applied to our problem.

B. Performance Analysis

MNMP is by design a greedy algorithm that incrementally places monitors and removes redundant monitors without backtracking. Generally, this greedy approach leads to suboptimal solutions for an arbitrary probing mechanism. We show, however, that for a probing mechanism of particular interest, CAP, MNMP provides the optimal solution.

Theorem 9: MNMP-CAP selects the minimum number of monitors for achieving k-identifiability under CAP.

Proof: See Appendix A. ■

Besides proving the optimality of MNMP-CAP, Theorem 9 also helps to establish fundamental bounds for other probing mechanisms. Since CAP is arguably the most powerful probing mechanism, the minimum number of monitors required by CAP, computed by MNMP-CAP, is a lower bound on the number of monitors required by any other probing mechanism. Specifically, let M^{CAP}, M^{CSP}, and M^{UP} denote the sets of monitors selected by MNMP-CAP, MNMP-CSP, and MNMP-UP, respectively, and μ^{CAP}, μ^{CSP}, and μ^{UP} be the minimum number of monitors under the corresponding probing mechanism. Then $\mu^{\text{CAP}} = |M^{\text{CAP}}|$, $|M^{\text{CAP}}| \le \mu^{\text{CSP}} \le |M^{\text{CSP}}|$, and $|M^{\text{CAP}}| \le \mu^{\text{UP}} \le |M^{\text{UP}}|$.

V. Improved Monitor Placement for Single-Failure Localization

An application of particular interest is localization of single-node failures. In practice, a monitor placement that uniquely localizes single-node failures allows the network operator to effectively handle *independent* failures, as there is at most one such failure (with high probability) at any point of time.

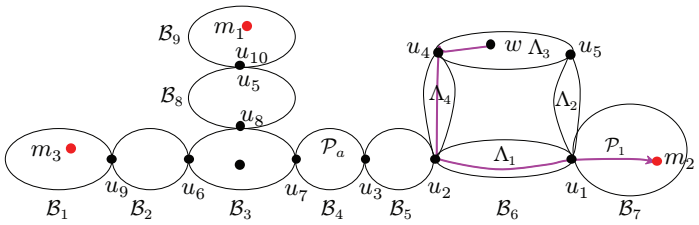


Fig. 4. Monitor placement in the sample network, where $\mathcal{B}_i$ ($i = 1, \ldots, 9$) is a biconnected component ($\{u_2, u_1\}$, $\{u_2, u_4\}$, $\{u_5, u_1\}$, $\{u_4, u_5\}$, $\{u_4, u_1\}$, and $\{u_2, u_5\}$ are 2-vertex-cuts in $\mathcal{B}_6$).

To localize single-node failures (i.e., achieve 1-identifiability), it suffices to ensure that any two failure sets, each containing a single non-monitor, are distinguishable (recall that monitor failure can be directly observed). We simplify Definition 2 into the following definition for 1-identifiability (P_v: set of measurement paths traversing node v, N: set of non-monitors).

Definition 10: A network $\mathcal{G}$ with measurement paths P is 1-identifiable if:

(1) $P_v \neq \emptyset$ for any $v \in N$, and

(2) $P_v \neq P_w$ for any $v, w \in N$ and $v \neq w$.

In Definition 10, the first condition guarantees that the failure of each non-monitor v is detectable (i.e., v is traversed by at least one measurement path), and the second condition guarantees that the observed path states uniquely localize the failure (if any). Ensuring these conditions using the minimum number of monitors, however, requires different approaches with highly different complexities under different probing mechanisms, as stated below.

Under CAP, Theorem 5 implies that as long as $\mathcal{G}$ has at least one monitor, we have $\Omega^{\text{CAP}}(v) = |C_{\mathcal{G}'}(v, m')| \geq 1$ for any node $v \in V$. Therefore, to achieve 1-identifiability under CAP, it suffices to place a single monitor at a randomly selected node. In contrast, UP complicates the problem by restricting paths to a predetermined set (i.e., routable paths). The problem of monitor placement thus becomes a combinatorial optimization that is hard to solve even for $k = 1$. To see this, we note that even if each monitor measures a predetermined set of paths that is independent of other monitors, the problem of placing the minimum number of monitors to satisfy conditions (1-2) in Definition 10 is NP-complete [7]. Monitor placement under UP is intuitively harder because the set of paths measured by a monitor depends on the locations of other monitors; a formal proof of the hardness is left to future work.

What remains open is whether the optimal monitor placement under CSP can be computed in polynomial time. In the rest of this section, we give a positive answer to this question by developing an algorithm that computes an optimal monitor placement for achieving 1-identifiability under CSP in linear time.

A. Observations

The proposed optimal monitor placement algorithm under CSP is motivated by the following three observations.

Observation 1. *Necessary monitor placement in biconnected subgraphs:* As discussed in Section IV-A2, to satisfy Condition 1 in Definition 10, each biconnected component with only one cut-vertex must select a non-cut-vertex as a monitor.

Observation 2. 1-*identifiable nodes after necessary monitor placement:* Assuming necessary monitors have been placed according to Observation 1, we prove that the network satisfies Condition 1 in Definition 10. Moreover, all nodes that are *neither cut-vertices nor nodes in* 2*-vertex-cuts* are 1-identifiable. Intuitively, this is because for each such node v (suppose v is not a monitor), $\exists$ a measurement path traversing any other non-monitor even if v is

removed from the network. For instance, after deploying monitors m_1, m_2, and m_3 according to Observation 1 in Fig. 4, if biconnected component $\mathcal{B}_7$ does not contain any internal 2-vertex-cuts, then all nodes in $V(\mathcal{B}_7) \setminus \{u_1\}$ are 1-identifiable ($V(\mathcal{B})$ denotes the set of nodes in graph $\mathcal{B}$), because for any nodes $z \in V(\mathcal{B}_7) \setminus \{u_1\}$, $\exists$ a path from m_2 to u_1 that does not traverse z.

Observation 3. *Additional monitor placement for achieving* 1*-identifiability:* We notice that after necessary monitor placement by Observation 1, cut-vertices and nodes in 2-vertex-cuts may or may not be 1-identifiable, in which case additional monitors may be required. For instance, consider the sample network in Fig. 4. Cut-vertices u_6, $u_7, \ldots, u_{10}$ are 1-identifiable. However, cut-vertex pairs $\{u_3, u_2\}$, $\{u_3, u_1\}$, $\{u_1, u_2\}$, and 2-vertex-cut node pair $\{u_4, u_5\}$ are not distinguishable. This is because for each of these node pairs, any simple path traversing one node also traverses the other. The optimal way to deploy additional monitor is to select a node in Λ_3 as a monitor, say w. Then measurement path $\mathcal{P}_1$ can be used to distinguish node pairs in $\{u_3, u_2\}$, $\{u_3, u_1\}$, and $\{u_4, u_5\}$; path $w \to u_5 \to u_1 \to m_2$ can distinguish between failures of u_1 and u_2.

Based on the above three observations, we are ready to develop the optimal monitor placement under CSP in a given network, for which we first sketch the basic idea as follows.

B. Algorithm Overview

We propose Optimal Monitor Placement for 1-identifiability under CSP (OMP-CSP), which consists of the following three basic steps:

(i) Ensure that all nodes that are neither cut-vertices nor nodes in 2-vertex-cuts are 1-identifiable by placing necessary monitors in biconnected components according to Observation 1;

(ii) Ensure that all nodes which are not cut-vertices but in 2-vertex-cuts are 1-identifiable by determining all 1-identifiable non-cut-vertices in each biconnected component according to Observation 2 and deploying additional monitors in each biconnected component according to Observation 3, such that all non-cut-vertices are 1-identifiable;

(iii) Ensure that all remaining nodes are 1-identifiable by placing the minimum number of additional monitors to make all cut-vertices 1-identifiable.

One special case for OMP-CSP is that the given network is already 2-connected, for which only step (ii) in OMP-CSP is required. We point out that the selected monitors in step (iii) do not affect the necessity of previously deployed monitors; see Theorem 17 for the proof. Overall, the monitors selected by OMP-CSP form an optimal solution (Theorem 17), i.e., any single-node failure can be uniquely localized using the minimum number of monitors.

C. Sketch of Algorithm

We now present a sketch of OMP-CSP and refer the reader to Appendix B for full detail. A basic object used in OMP-CSP is the biconnected/triconnected component defined in Definition 8. Intuitively, a triconnected component within a biconnected component is connected to the rest of the biconnected component by 2-vertex-cuts. However, not all subgraphs within a biconnected component separated by 2-vertex-cuts form triconnected components according to Definition 8. For example, $\mathcal{D}_1$ in Fig. 3 (b) has a 2-vertex-cut $\{b, c\}$; however, $\mathcal{D}_1$ is not 3-connected. To fix this issue, we add *virtual links* to the graph as follows: For each 2-vertex-cut whose vertices are not neighbors, we connect the vertices by a virtual link; we repeat this procedure for all 2-vertex-cuts. It can be verified that all subgraphs separated by cut-vertices and 2-vertex-cuts in the processed graph are triconnected components. For

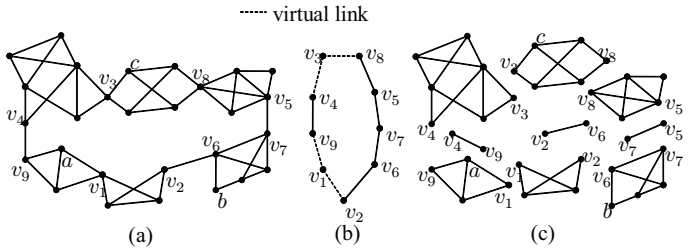


Fig. 5. PLC decomposition: (a) Sample 2-connected network ($M = \{a, b, c\}$ is the optimal monitor placement for achieving 1-identifiability; see Appendix B for details); (b) Polygon in (a); (c) All PLCs in (a).

instance, Fig. 3 (c) shows the triconnected components, separated by cut-vertices and 2-vertex-cuts $\{b, c\}$, $\{i, f\}$, $\{i, q\}$, and $\{f, l\}$.

One can compute all the triconnected components of a graph in linear time by the graph decomposition algorithm in [22]. According to Observation 2, placing necessary monitors by Observation 1 ensures that all internal nodes in triconnected components (i.e., neither cut-vertices nor nodes in 2-vertex-cuts) are 1-identifiable. However, this observation does not capture all 1-identifiable nodes, as there may be other nodes that are also 1-identifiable (see Theorem 15). To address this issue, we introduce a new type of graph decomposition built upon the triconnected component decomposition as follows.

PLC decomposition: Our proposed decomposition divides a graph into certain subgraphs called *polygon-less components (PLCs)*, which are defined based on the concepts of *merging* and *polygon* given below.

Definition 11: If two components $\mathcal{T}_1 = (V_1, L_1)$ and $\mathcal{T}_2 = (V_2, L_2)$ share the same virtual link vw and no other component shares vw, then $\mathcal{T}_1$ and $\mathcal{T}_2$ are *mergeable*. The merged component equals the union of the two components without the virtual link, i.e., $\mathcal{T} = (V_1 \cup V_2, L_1 \cup L_2 \setminus vw)$.

Definition 12: After merging all mergeable triangles generated from triconnected component decomposition of $\mathcal{G}$, each merged component with at least four nodes is called a *polygon* of $\mathcal{G}$.

For example, in Fig. 3 (c), two triangles $(\{i, q, f\}, \{iq, if, qf\})$ and $(\{i, f, c\}, \{if, ic, cf\})$ can be merged to form a quadrilateral. However, virtual link fl in Fig. 3 (c) is the common link among three triangles; therefore, triangles $(\{f, g, l\}, \{fg, fl, gl\})$, $(\{f, k, l\}, \{fk, fl, kl\})$, and $(\{f, o, l\}, \{fo, fl, ol\})$ in Fig. 3 (c) cannot be merged. For the example in Fig. 5, Fig. 5 (b) is a polygon in Fig. 5 (a).

Intuitively, a graph containing a polygon can be decomposed into subgraphs separated by vertices of the polygon. We refer to such subgraphs as PLCs, defined as follows.

Definition 13: A Polygon-Less Component (PLC) is a maximal subgraph of $\mathcal{G}$ that is either (i) a 2-connected graph containing at most two vertices in any polygon of $\mathcal{G}$, or (ii) a complete graph with 2 vertices (i.e., bonds).

For instance, Fig. 5 (c) shows all PLCs (including bonds) in Fig. 5 (a). As the name suggests, PLCs do not contain any polygons. Moreover, Fig. 5 shows that, as opposed to triconnected components, PLCs do not contain any virtual links.

Note that the set of triconnected components found by the graph decomposition algorithm in [22] is not unique, as it depends on the order in which virtual links are added. In contrast, PLC decomposition has a desirable property that its outcome is unique, as stated in the following lemma, proved in Appendix C.

Lemma 14: Any network $\mathcal{G}$ can be uniquely decomposed into a set of PLCs.

The significance of PLC is that it makes it easy to determine all 1-identifiable nodes in a PLC, even if this PLC may contain 2-vertex-cuts; see Theorem 15 (proved in Appendix D).

Algorithm 2: Optimal Monitor Placement for 1-identifiability under CSP (OMP-CSP)

input : Connected network topology $\mathcal{G}$ which is not 2-connected

output: Set of monitors that achieves 1-identifiability in $\mathcal{G}$ under CSP

1 partition $\mathcal{G}$ into biconnected components $\{\mathcal{B}_1, \mathcal{B}_2, \ldots\}$ and then PLCs;
2 **if** $\mathcal{G}$ *is 2-connected* **then**
3 deploy monitors by the auxiliary algorithm *Monitors-in-Biconneted-Network* (see Appendix B);
4 return;
5 **end**
6 **foreach** *biconnected component* $\mathcal{B}_i$ **do**
7 **if** $\mathcal{B}_i$ *is a PLC* ***and*** $\mathcal{B}_i$ *has only one cut-vertex* **then**
8 randomly choose node v (v is not a cut-vertex in $\mathcal{G}$) in $\mathcal{B}_i$ as a monitor;
9 **else**
10 find set A containing all PLCs with ≥ 3 agents or ≥ 4 neighboring PLCs within $\mathcal{B}_i$, and set C containing all neighboring PLCs of each PLC in set A within $\mathcal{B}_i$;
11 within $\mathcal{B}_i$, find set E containing all PLCs with only 2 agents and one agent is a cut-vertex (in $\mathcal{G}$);
12 $\mathcal{B}_i' \leftarrow \mathcal{B}_i \ominus (A \cup C \cup E)$;
13 *Monitors-in-Polygon-less-Network*($\mathcal{B}_i'$, S_a^i), where S_a^i is the set of agents in $\mathcal{B}_i$;
14 **end**
15 **end**
16 find set F containing all biconnected components with monitors;
17 find set I containing all biconnected components with 2 cut-vertices (in $\mathcal{G}$) and 3 or more neighboring biconnected components in $\mathcal{G}$;
18 find set J containing all biconnected components with 3 or more cut-vertices, and set K containing all neighboring biconnected components of each component in J;
19 $\mathcal{G}' \leftarrow \mathcal{G} \ominus (F \cup I \cup J \cup K)$;
20 *Monitors-in-Polygon-less-Network*($\mathcal{G}'$, S_c), where S_c is the set of cut-vertices in $\mathcal{G}$;

Theorem 15: If a PLC contains at least two monitors, then all nodes in this PLC are 1-identifiable.

Moreover, we have Corollary 16, proved in Appendix E, for determining the identifiable nodes in a PLC without monitors.

Corollary 16: Let Λ be a PLC in $\mathcal{G}$. If Λ contains two cut-vertices of $\mathcal{G}$, then all its non-cut-vertices are 1-identifiable; if Λ contains three or more cut-vertices of $\mathcal{G}$, then all its nodes are 1-identifiable.

Corollary 16 implies that cut-vertices can be considered as *effective monitors* for localizing node failures in PLCs. Moreover, nodes on polygons can also assist in identifying failed nodes in PLCs (see the proof of Theorem 17). In this regard, we refer to both cut-vertices and nodes on polygons as *agents* for localizing node failures in the sequel.

Algorithm OMP-CSP: With the above definitions, we now present OMP-CSP (Algorithm 2), which follows the three logical steps outlined in Section V-B.

(1) First, we decompose $\mathcal{G}$ into biconnected components and then each biconnected component into PLCs (line 1). Based on this decomposition, we place monitors according to Observation 1 in each biconnected component that is a PLC with only one cut-vertex (line 8). In cases where the given network is already 2-connected (i.e., containing a single biconnected component), Observation 1 cannot be applied; instead, we select monitors by an auxiliary algorithm *Monitors-in-Biconneted-Network*, given in Algorithm A in Appendix B (line 3). The basic idea of Algorithm A is to first remove all nodes that are 1-identifiable (determined by the multiple-polygon structure (line 5 of Algorithm A) or the necessary initial monitor placement (lines 10 and 21 of Algorithm A)), such that the

Algorithm 3: Monitors-in-Polygon-less-Network($\mathcal{G}$, S)

input : Network topology $\mathcal{G}$, node set S

output: Subset of nodes in $\mathcal{G}$ as monitors

1 **if** $|L| = 0$ **then**
2 return;
3 **end**
4 **foreach** *connected component* $\mathcal{G}_i$ *in* $\mathcal{G}$ **do**
5 **if** $\mathcal{G}_i$ *contains only one biconnected component* **then**
6 randomly choose a node in $\mathcal{G}_i$ as a monitor;
7 **else**
8 in $\mathcal{G}_i$, label one biconnected component with 0 or 1 cut-vertex as $\mathcal{B}_1$, one neighboring biconnected component of $\mathcal{B}_1$ as $\mathcal{B}_2$ (if any), and one neighboring biconnected component of $\mathcal{B}_2$ other than $\mathcal{B}_1$ as $\mathcal{B}_3$ (if any);
9 **if** $\mathcal{B}_2$ *is a bond* **then**
10 choose the common node between $\mathcal{B}_1$ and $\mathcal{B}_2$ as a monitor;
11 $\mathcal{G}_i' \leftarrow \mathcal{G}_i \ominus (\mathcal{B}_1 + \mathcal{B}_2)$;
12 **else** `//`$\mathcal{B}_2$ `is not a bond`
13 randomly choose node v ($v \notin S$) in $\mathcal{B}_2$ as a monitor;
14 $\mathcal{G}_i' \leftarrow \mathcal{G}_i \ominus (\mathcal{B}_1 + \mathcal{B}_2 + \mathcal{B}_3)$ (if $\mathcal{B}_3$ exists);
15 **end**
16 *Monitors-in-Polygon-less-Network*($\mathcal{G}_i'$, S);
17 **end**
18 **end**

remaining graph does not contain any polygons. Then for this polygon-less graph, we use Algorithm 3 (see step (3)) for additional monitor placement. If the input graph is not 2-connected, then the following two steps are executed.

(2) Next, we process the rest of the biconnected components. We prove in [23] that using the monitors placed by Observation 1, if a biconnected component contains non-cut-vertices that are not 1-identifiable, then this biconnected component must have at least one polygon and additional monitors are required to be *inside* this biconnected component for achieving network 1-identifiability. Therefore, the goal of this step is to deploy additional necessary monitors to ensure that each non-cut-vertex is 1-identifiable. We address this issue by finding all PLCs containing non-cut-vertices that are not 1-identifiable in each biconnected component according to Observation 2 (lines 10–12) via a graphical operation "$\ominus$" (line 12) defined as follows: Let $\mathcal{G}_2$ be a subgraph of $\mathcal{G}_1$ and V' the set of common nodes between $\mathcal{G}_1$ and $\mathcal{G}_2$. Then $\mathcal{G}_1 \ominus \mathcal{G}_2$ denotes a subgraph of $\mathcal{G}_1$ after removing all nodes in $\mathcal{G}_2$ except for nodes in V'. To ensure 1-identifiability in the remaining graph ($\mathcal{B}_i'$ in line 12), we use the auxiliary algorithm *Monitors-in-Polygon-less-Network* (Algorithm 3) in line 13 to further deploy monitors according to Observation 3. The details of Algorithm 3 will be explained in step (3).

(3) After placing monitors according to the above two steps, we now determine the biconnected components with cut-vertices that are not 1-identifiable (lines 16–19). Unlike step (2) where we place monitors inside a biconnected component to ensure that all its non-cut-vertices are 1-identifiable, the optimal monitor placement for identifying cut-vertices in a biconnected component may be *outside* this biconnected component (see the proof of Theorem 17). To place these monitors optimally, we propose a recursive algorithm in Algorithm 3. The idea of Algorithm 3 is to select one monitor per recursion such that the number of 1-identifiable cut-vertices is maximized. We prove that, up to three cut-vertices can become 1-identifiable by placing one additional monitor; moreover, these three cut-vertices

belong to three neighboring biconnected components and the monitor must be placed at an internal node (non-cut-vertex) of the middle component (see the proof of Theorem 17). Therefore, lines 8–16 in Algorithm 3 examine if there exists such a monitor placement for three neighboring biconnected components ($\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ in line 8). Note that if the middle biconnected component ($\mathcal{B}_2$) is a bond (recall that a bond consists of two nodes connected by one link), then it does not contain any internal node. This special case is handled by line 10.

Remark: In line 12 of Algorithm 2, the PLCs of $\mathcal{B}'_i$ essentially are also biconnected components of $\mathcal{B}'_i$ (i.e., similar to $\mathcal{G}'$ in line 19 of Algorithm 2, $\mathcal{B}'_i$ contains no polygons); therefore, Algorithm 3 can also be used for processing $\mathcal{B}'_i$ (in line 13 of Algorithm 2). As an example in Fig. 4, after placing monitors m_1–m_3 by line 8 in Algorithm 2, the input graph to Algorithm 3 is Λ_3 (Λ_3 is a PLC of $\mathcal{B}_6$). In this case, non-cut-vertex w is selected as a monitor by line 6 of Algorithm 3 whereby any two nodes in $\{u_1, u_2, u_3, u_4, u_5\}$ are distinguishable.

Example: Fig. 3 illustrates one example. The optimal monitor placement for achieving 1-identifiability in Fig. 3 consists of a, g, and h as monitors, where a and g are selected by line 8 in Algorithm 2, while h is selected by line 6 in Algorithm 3.

D. Performance Analysis

Optimality: The correctness and the optimality of OMP-CSP are shown in Theorem 17; see [23] for the proof.

Theorem 17: OMP-CSP ensures that any single-node failure in a given network is uniquely identifiable under CSP using the minimum number of monitors.

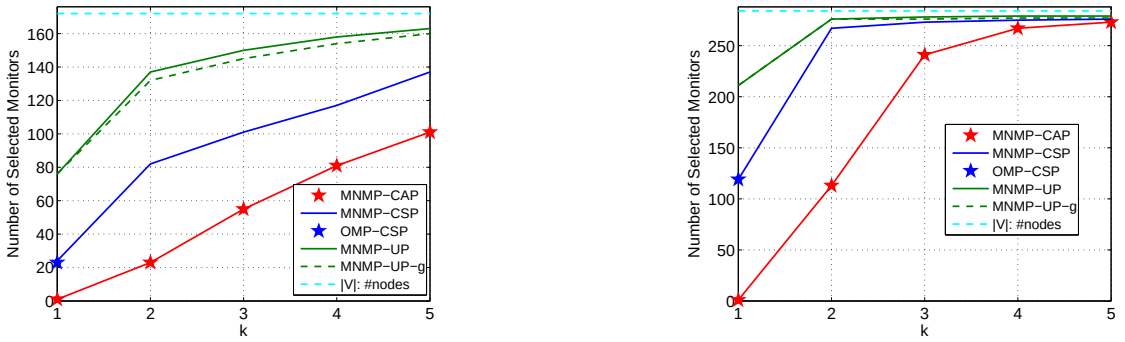
Complexity: In OMP-CSP, splitting $\mathcal{G}$ into biconnected and then triconnected components takes $O(|V|+|L|)$ time [21], [22]. Using these triconnected components, OMP-CSP further takes $O(|T|)$ time to find all PLCs, where T is the set of triconnected components. Processing all biconnected components in lines 6–19 takes $O(|T|+|B|)$ time, where B is the set of biconnected components. In addition, Algorithm 3 exhibits $O(|B_I|)$ complexity, where B_I is the set of biconnected components of the input network for Algorithm 3. Moreover, the time complexity of Algorithm A (see Appendix B) is $O(|T|)$. Therefore, the entire algorithm of OMP-CSP has $O(|V|+|L|)$ time complexity.

Discussion: A previous study [24] developed an optimal monitor placement algorithm for localizing single-link failures under controllable "m-trail" probing (allowing repeated nodes but not repeated links in measurement paths), based on a similar approach that decomposes the network into 2/3-edge-connected subgraphs and places monitors in each subgraph sequentially. Although it is possible to cast node failures to link failures by transforming the network topology $\mathcal{G}$ (e.g., representing each node by two logical nodes connected by a logical link), the transformation introduces links not of interest (not representing nodes in $\mathcal{G}$) and paths not measurable under CSP (e.g., 2-hop path from a monitor to its neighbor and back). Thus, the solution in [24] cannot be applied to place monitors for localizing single-node failures under CSP.

VI. Impact of Probing Mechanisms on Monitor Placement

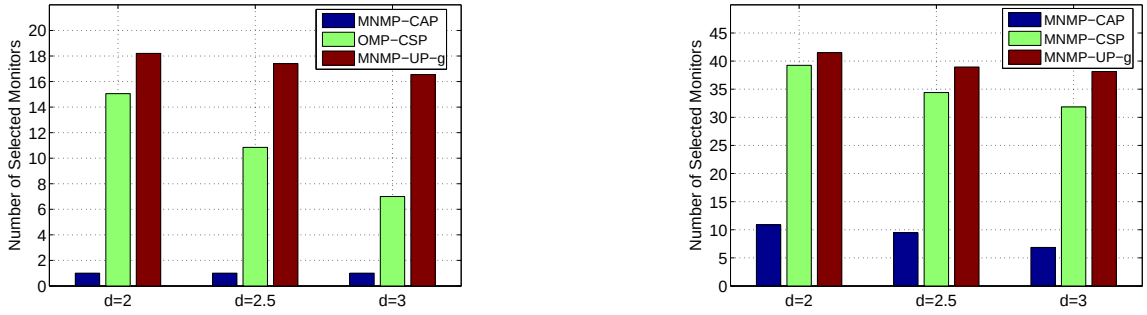
Given the above results, we are now ready to quantify the impact of the probing mechanism on monitor placement. We aim to quantify this impact by evaluating the minimum number of monitors required for achieving k-identifiability under each of the three probing mechanisms (CAP, CSP, UP). In this study, we assume shortest (hop-count) path routing as the default routing protocol under UP, with ties broken arbitrarily.

Topologies for Evaluation: We first evaluate the monitor requirement on real *Autonomous System* (AS) topologies collected by the Rocketfuel [25] and the CAIDA [26] projects, which represents IP-level connections between



(a) Rocketfuel AS1755 ($|V| = 172$, $|L| = 381$) (b) CAIDA AS28583 ($|V| = 284$, $|L| = 415$)

Fig. 6. Monitor placement in AS topologies (★: optimal value; MSC(v) is computed by enumeration in MNMP-UP; GSC(v) is used to approximate MSC(v) in MNMP-UP-g).



(a) $k = 1$ (b) $k = 2$

Fig. 7. Monitor placement in ER graphs ($|V| = 50$, d is the average node degree, $\mathbb{E}[|L|] =$d$|V|/2$, 20 graph realizations for each d).

backbone/gateway routers of several ASes from major *Internet Service Providers (ISPs)* around the globe. The selected AS topologies for evaluations are Rocketfuel AS1755 with 172 nodes and 381 links, and CAIDA AS28583 with 284 nodes and 415 links. We then test the monitor requirement on a comprehensive set of randomly generated topologies without artifacts of specific network deployments. We consider random Erdös-Rényi (ER) graphs [27], generated by independently connecting each pair of nodes by a link with a fixed probability p. The result is a purely random topology where all graphs with an equal number of links are equally likely to be selected (note that the number of nodes is an input parameter).

Tightness of Bounds: The computation of maximum node identifiability $\Omega(v)$ is at the core of our monitor placement algorithm. Under CAP and CSP, we can accurately characterize $\Omega^{\text{CAP}}(v)$ and $\Omega^{\text{CSP}}(v)$ (see Sections III-B, III-C); under UP, we only have a loose bound on $\Omega^{\text{UP}}(v)$ due to the hardness in evaluating MSC(v) (Section III-D). Nevertheless, [13] has empirically verified that GSC(v) closely approximates MSC(v) in a variety of network topologies. We have also verified in Fig. 6 that the number of monitors placed by MNMP-UP based on GSC(v) ('MNMP-UP-g') closely approximates that based on MSC(v) ('MNMP-UP'). These results suggest that for monitor placement under UP, we can use GSC(v) in place of MSC(v) for computing a tight lower bound on $\Omega^{\text{UP}}(v)$.

Evaluation Results: The result of monitor placement in one Rocketfuel AS toplogy is reported[3] in Fig. 6 (a)[4]. The result shows a clear difference in the monitor requirement under different probing mechanisms and different scales of failures (i.e., different values of k). Specifically, while CAP can achieve 1-identifiability using a single

[3]For monitor placement under CSP and UP, if the computed maximum node identifiability is $0 \leq \Omega(v) \leq 1$, then to reduce the total number of selected monitors, exhaustive examination of the two conditions in Definition 10 is used to determine if $\Omega(v)$ is 0; if not, then $\Omega(v) = 1$.

[4]Similar results have been observed for other ASes in this Rocketfuel dataset, which are omitted in the paper due to space limitations.

monitor, CSP and UP require a substantial fraction (0.13 for CSP, and 0.46 for UP) of nodes as monitors for achieving 1-identifiability in the same network topology. This result demonstrates substantial resource saving in localizing node failures by leveraging controllable probing mechanisms. When k increases, a considerable number of additional monitors is required, especially for UP, where most nodes in the network should be monitors for localizing multi-node failures (i.e., $k \geq 2$). In addition, when $k = 1$, Fig. 6 (a) shows that the monitor placement computed by the greedy algorithm MNMP-CSP closely approximates the optimal placement obtained by OMP-CSP; note that OMP-CSP is still preferred in selecting monitors for 1-identifiability, as OMP-CSP is provably optimal and only has linear time complexity. Fig. 6 (a) also demonstrates that with a fixed number of monitors, the network capability in localizing multiple failures can be improved by relaxing probing constraints. Specifically, by selecting 80 monitors, we can only achieve 1-identifiability under UP; by contrast, 2-identifiability can be achieved under CSP, and even 4-identifiability under CAP. Furthermore, as MNMP-CAP computes the minimum monitor requirement under the most flexible routing mechanism that can possibly be allowed in communication networks, the number of monitors placed by MNMP-CAP establishes a lower bound for any other probing mechanism. For instance, 58% of nodes are required to be monitors under CAP for $k = 5$, which suggests that to uniquely localize failures of up to 5 nodes, we need at least 58% of nodes to be monitors regardless of the probing mechanism.

Because ISP topologies have evolved since the Rocketfuel project, we repeat the above evaluation on a recent dataset obtained by the CAIDA project; see results in Fig. 6 (b). In Fig. 6 (b), we first draw similar conclusions as those in Fig. 6 (a). In addition, compared with the Rocketfuel AS, we observe that the CAIDA AS requires more monitors under the same network settings (except for 1-identifiability under CAP), e.g., over 95% of nodes are required to be monitors under CSP and UP for $k \geq 2$. Moreover, relaxing the probing mechanism from UP to CAP only slightly reduces the required number of monitors for $k \geq 4$. This is because the network used in Fig. 6 (b) is sparser compared to the network in Fig. 6 (a), as indicated by a smaller average node degree (i.e., $2|L|/|V|$).

To further examine how the network link density affects the monitor requirement under different probing mechanisms, we tune the average node degree in random ER graphs and compare the corresponding number of selected monitors. In this evaluation, we fix the number of nodes to 50, and randomly generate 20 graph realizations[5] for each given average node degree, which are then fed to monitor selection algorithms. The monitor requirement averaged over multiple graph realizations are reported in Fig. 7. Clearly, Fig. 7 confirms the intuition that poorly-connected networks require more monitors to localize failures due to their limited choices of probing paths. Note that CAP remains superior for achieving k-identifiability in both real and synthetic network topologies for a small k, as it allows probes to be sent and received by the same monitor.

VII. Conclusion

We investigated the problem of localizing failed nodes from binary states of end-to-end paths between monitors, focusing on the problem of placing the minimum number of monitors to identify a given number of failures. We studied this problem by first establishing a fine-grained measure of failure identifiability, based on which we propose a generic monitor placement algorithm that is applicable under any probing mechanism. We then concretized the proposed algorithm for three representative probing mechanisms, which offer different tradeoffs between the controllability of probe routing and the complexity of implementation. We proved that the proposed algorithm achieves the minimum number of monitors for a probing mechanism that allows monitors to measure arbitrary

[5]All these realizations are checked before use to ensure they are connected.

paths (subject to connectivity), and provides upper/lower bounds for the other probing mechanisms. In the special case of single-node failures, we developed an improved algorithm that minimizes the number of monitors when measurement paths must be cycle-free. Our evaluations on real network topologies reveal that giving the monitors more control over the routing of probes can substantially reduce the number of monitors required for localizing node failures, or improve the number of failures that can be simultaneously localized using a given number of monitors, especially in well-connected networks.

REFERENCES

[1] R. Kompella, J. Yates, A. G. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes," in *IEEE INFOCOM*, 2007.

[2] N. Duffield, "Simple network performance tomography," in *IMC*, 2003.

[3] D. Ghita, C. Karakus, K. Argyraki, and P. Thiran, "Shifting network tomography toward a practical goal," in *ACM CoNEXT*, 2011.

[4] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, vol. 52, pp. 5373–5388, 2006.

[5] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in *IEEE INFOCOM*, 2003.

[6] H. X. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in IP networks," in *PAM*, 2004.

[7] S. Stanic, S. Subramaniam, G. Sahin, H. Choi, and H.-A. Choi, "Active monitoring and alarm management for fault localization in transparent all-optical networks," *IEEE Transactions on Network and Service Management*, vol. 7, pp. 118–131, 2010.

[8] H. Nguyen and P. Thiran, "The boolean solution to the congested IP link location problem: Theory and practice," in *IEEE INFOCOM*, 2007.

[9] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *ACM CoNEXT*, 2007.

[10] Y. Huang, N. Feamster, and R. Teixeira, "Practical issues with using network tomography for fault diagnosis," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 53–58, 2008.

[11] S. Ahuja, S. Ramasubramanian, and M. Krunz, "SRLG failure localization in all-optical networks using monitoring cycles and paths," in *IEEE INFOCOM*, 2008.

[12] S. Cho and S. Ramasubramanian, "Localizing link failures in all-optical networks using monitoring tours," *Elsevier Computer Networks*, vol. 58, pp. 2–12, 2014.

[13] L. Ma, T. He, A. Swami, D. Towsley, K. K. Leung, and J. Lowe, "Node failure localization via network tomography," in *ACM IMC*, 2014.

[14] M. Cheraghchi, A. Karbasi, S. Mohajer, and V. Saligrama, "Graph-constrained group testing," *IEEE Transactions on Information Theory*, vol. 58, pp. 248–262, 2012.

[15] M. Wang, W. Xu, E. Mallada, and A. Tang, "Network tomography via sparse recovery," *IEEE Transactions on Information Theory*, vol. 61, pp. 1028–1044, 2015.

[16] J. Chuzhoy and S. Khanna, "Polynomial flow-cut gaps and hardness of directed cut problems," *Journal of the ACM*, vol. 56, pp. 1–28, 2009.

[17] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[19] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, pp. 233–235, 1979.

[20] R. Diestel, *Graph theory*. Springer-Verlag Heidelberg, New York, 2005.

[21] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, pp. 146–160, 1972.

[22] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM Journal on Computing*, vol. 2, pp. 135–158, 1973.

[23] L. Ma, T. He, A. Swami, D. Towsley, and K. K. Leung, "Node failure localization: Theorem proof," Technical Report, April 2015. [Online]. Available: http://researcher.watson.ibm.com/researcher/files/us-maliang/NodeFailureTechReportApr15.pdf

[24] S. Ahuja, S. Ramasubramanian, and M. Krunz, "Single-link failure detection in all-optical networks using monitoring cycles and paths," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1080–1093, 2009.

[25] "Rocketfuel: An ISP topology mapping engine," University of Washington, 2002. [Online]. Available: http://www.cs.washington.edu/research/networking/rocketfuel/

[26] "Macroscopic Internet Topology Data Kit (ITDK)," The Cooperative Association for Internet Data Analysis (CAIDA), April 2013. [Online]. Available: http://www.caida.org/data/active/internet-topology-data-kit/

[27] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, pp. 17–61, 1960.

APPENDIX

Table I summarizes all graph-theoretical notions used in this Appendix (following the convention in [20]).

TABLE I
GRAPH-RELATED NOTATIONS

Symbol	Meaning
V, L	set of nodes/links
M, N	set of monitors/non-monitors ($M \cup N = V$)
$V(\mathcal{G})$, $L(\mathcal{G})$	set of nodes/links in graph $\mathcal{G}$
$\|\mathcal{G}\|$	$\|\mathcal{G}\| = \|V(\mathcal{G})\|$
$\mathcal{G} - L'$	delete links: $\mathcal{G} - L' = (V, L \setminus L')$, where "\" is setminus
$\mathcal{G} + L'$	add links: $\mathcal{G} + L' = (V, L \cup L')$, where the end-points of links in L' must be in V
$\mathcal{G} + \mathcal{G}'$	combine two graphs: $\mathcal{G} + \mathcal{G}' = (V(\mathcal{G}) \cup V(\mathcal{G}'), L(\mathcal{G}) \cup L(\mathcal{G}'))$
$L(v)$	set of links incident to node v
$\mathcal{G} - v$	delete a node: $\mathcal{G} - v = (V(\mathcal{G}) \setminus \{v\}, L(\mathcal{G}) \setminus L(v))$, where $v \in V(\mathcal{G})$
$\mathcal{G}_s + v$	add a node: $\mathcal{G}_s + v = (V(\mathcal{G}_s) \cup \{v\}, L(\mathcal{G}_s) \cup L_v)$, where $\mathcal{G}_s$ is a subgraph of $\mathcal{G}$, $v \in V(\mathcal{G}) \setminus V(\mathcal{G}_s)$, and L_v is the set of links in $L(\mathcal{G}) \setminus L(\mathcal{G}_s)$ that connect v and nodes in $V(\mathcal{G}_s)$
$\mathcal{G} - S$	delete a node set: $\mathcal{G} - S = \mathcal{G} - \sum_{i=1}^{\|S\|} s_i$, where $S \subseteq V(\mathcal{G})$ and $S = \{s_i\}_{i=1}^{\|S\|}$
$\mathcal{G}_s + S$	add a node set: $\mathcal{G}_s + S = \mathcal{G}_s + \sum_{i=1}^{\|S\|} s_i$, where $\mathcal{G}_s$ is a subgraph of $\mathcal{G}$, $S \subseteq V(\mathcal{G}) \setminus V(\mathcal{G}_s)$ and $S = \{s_i\}_{i=1}^{\|S\|}$

A. *Proof of Theorem 9*

To prove the optimality of MNMP-CAP, we first introduce another monitor placement scheme, called Structural Monitor Placement under CAP (SMP-CAP), which is provable to be optimal in reducing the total number of monitors under CAP. Our idea is to show that MNMP-CAP and SMP-CAP select the same number of monitors, thus proving the optimality of MNMP-CAP. For SMP-CAP, it is based on the following definitions.

Definition 18: The *split operation* on a q-connected graph $\mathcal{D}$ ($|\mathcal{D}| \geq q+2$) with a q-vertex-cut C is to first get a collection of connected components by removing C, and then add C back to each of the connected components in this collection, i.e., after the split operation, $\mathcal{D}$ becomes $\{\mathcal{D}_i + C\}_{i=1}^{u}$, where $\{\mathcal{D}_i\}_{i=1}^{u} := \mathcal{D} - C$ ($\mathcal{D}_i$ is a connected component in $\mathcal{D} - C$, and u is the number of connected components in $\mathcal{D} - C$).

Definition 19: Apply the split operation to $\mathcal{G}$ and all of its resulting connected graphs recursively, until each generated graph is k-connected or a clique with up to k nodes. Such k-connected graph or clique is called a k-component.

For any set of k-components $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_z\}$ ($\mathcal{D}_i$ is a k-component), if its union $\mathcal{D}' := \sum_{i=1}^{z} \mathcal{D}_i$ (see Table I for notations) is a connected graph, then $\mathcal{D}'$ is called *k-concatenated component*, and the number of k-components involved in $\mathcal{D}'$ is called the *size* of $\mathcal{D}'$. According to the definition, a k-concatenated component in $\mathcal{G}$ can be as small as a single k-component, and as large as the entire $\mathcal{G}$. In a k-concatenated component $\mathcal{D}$ (within $\mathcal{G}$), the set of

Algorithm 4: Structural Monitor Placement under CAP (SMP-CAP)

input : Network topology $\mathcal{G}$, parameter k ($1 \leq k \leq |V|$)

output: Set of monitors $M \subseteq V$

1 **for** *layer* $q = 1, 2, \ldots, \sigma$ *(σ : the total number of k-components in $\mathcal{G}$)* **do**
2 **for** *each k-concatenated component $\mathcal{D}$ with q and only q k-components (size q)* **do**
3 $S \leftarrow \{$monitors or separation vertices in $\mathcal{D}\}$;
4 **if** $|S| < \min(k, |\mathcal{D}|)$ **then**
5 $M \leftarrow M \cup \{\min(k, |\mathcal{D}|) - |S|$ randomly selected nodes in $\mathcal{D}$ outside $S\}$;
6 **end**
7 **end**
8 **end**

nodes separating $\mathcal{D}$ from the rest of $\mathcal{G}$ are called *separation vertices* w.r.t. $\mathcal{D}$. With these definitions, we can present SMP-CAP, shown in Algorithm 4.

SMP-CAP first groups k-components of $\mathcal{G}$ into σ layers (σ : the total number of k-components in $\mathcal{G}$), with each layer consisting of k-concatenated components with fixed size, and then selects monitors structurally following the order from a lower layer to an upper layer, where a higher layer is indicated by a larger size of k-concatenated components it involves. Specifically, SMP-CAP starts selecting monitors from layer 1 (which contains all k-components), then increases the layer level (line 1 of SMP-CAP), and terminates at layer σ (which only contains the entire network $\mathcal{G}$). In each layer, SMP-CAP selects monitors as necessary to ensure that: (i) each involved k-concatenated component of size at least k has k nodes that are monitors or separation vertices; (ii) each involved k-concatenated component of size smaller than k has all the nodes as either monitors or separation vertices (line 5). We prove the optimality of SMP-CAP in Lemma 20.

Lemma 20: For any k ($1 \leq k \leq |V|$), SMP-CAP places a minimum set of monitors for achieving k-identifiability under CAP.

Proof: It is straightforward to verify that SMP-CAP places the minimum number of monitors to ensure that each k-concatenated component $\mathcal{D}$ has at least $\min(k, |\mathcal{D}|)$ nodes that are monitors or separation vertices, without which there exist nodes that are disconnected to monitors after removing all separation nodes and monitors (together the total number is less than $\min(k, |\mathcal{D}|)$) in $\mathcal{D}$.

Next, we focus on proving that SMP-CAP can achieve k-identifiability. We prove it by verifying the condition in Theorem 4. Suppose that we remove an arbitrary set of nodes S with $|S| \leq k-1$ (S may or may not contain monitors) from $\mathcal{G}$. Then we examine if a non-monitor v in $\mathcal{G}-S$ still connects to monitors. Let $\mathcal{U}$ denote the connected component that contains v in $\mathcal{G}-S$, $\mathcal{U}'$ the connected component that contains v in $\mathcal{U}+S$, and $\{\mathcal{D}_i\}_{i=1}^{\sigma}$ the set[6] of all k-components in $\mathcal{G}$. Since $|S| \leq k-1$ and each $\mathcal{D}_i$ in $\{\mathcal{D}_i\}_{i=1}^{\sigma}$ is either k-connected or a clique, S cannot separate any k-components. Therefore, $\mathcal{U}'$ is a k-concatenated component that involves v. Let $S' := S \cap V(\mathcal{D}')$, i.e., nodes in $S \setminus S'$ are in other k-concatenated components. Then all separation vertices w.r.t. $\mathcal{U}'$ are in S', because $\mathcal{U}$ is separated from the rest of the graph after removing S from $\mathcal{G}$. Since $v \notin S'$, v must be a non-separation vertex in $\mathcal{U}'$. As v is a non-monitor, we know that $|\mathcal{U}'| > k$ according to the monitor selection rule in line 5 of SMP-CAP. Let β be the number of separation vertices in $\mathcal{U}'$. As $\mathcal{U}$ is separated from the rest of the graph by removing node set S, we must have $\beta \leq |S'| \leq k-1$. Thus, at least $k-\beta$ non-separation vertices in $\mathcal{U}'$ (we have shown that

[6]Whether $\{\mathcal{D}_i\}_{i=1}^{\sigma}$ is unique or not does not affect the proof.

$|\mathcal{U}'| > k$ in a previous argument) should be selected as monitors. Suppose in addition to the separation vertices in S', S' also involves all monitors in $\mathcal{U}'$, i.e., v does not connect to any monitors in the connected graph $\mathcal{U}$. Then we have $|S'| - \beta \geq k - \beta$, and thus $|S'| \geq k$, contradicting the fact that $|S'| \leq k - 1$. Hence, S' cannot contain all monitors in $\mathcal{U}'$. Therefore, non-monitor v connects to at least one monitor in connected graph $\mathcal{U}$, i.e., $\mathcal{U}' - S'$. Consequently, for any node set S with $|S| \leq k - 1$, each connected component in $\mathcal{G} - S$ contains at least one monitor, i.e., $\Omega^{\text{CAP}}(v) \geq k$ for $v \in V$, thus confirming that SMP-CAP achieves k-identifiability. ■

With SMP-CAP, now we can prove the optimality of MNMP-CAP. MNMP-CAP guarantees that $\Omega(v) \geq k$ for each $v \in V(\mathcal{G})$, thus achieving k-identifiability. Therefore, it suffices to prove that the total number of monitors selected by MNMP-CAP cannot be further reduced.

(1) We first prove that for any subset M' in the monitor set M selected by MNMP-CAP, $M \setminus M'$ cannot achieve k-identifiability. Line 13 of MNMP-CAP examines redundant monitors in M in an arbitrary order. Suppose $m \in M$ is examined to be unremovable from M. In this case, it implies that m is required in at least one k-concatenated component even though some other nodes in the network may already be selected as monitors. Therefore, if we further remove some monitors from M, then m is still required in that k-concatenated component. Hence, m is guaranteed to be non-redundant even if later some other monitors in M are examined to be redundant, and thus there is no need to examine the redundancy of m again after removing some other monitors in M. Therefore, after the examination of line 13, each monitor in M is unremovable. Thus, as a subset of M, M' cannot be removed from M while maintaining k-identifiability.

(2) Next, we prove that for any subset $M' \subseteq M$ ($M' \neq \emptyset$, M is the monitor set selected by MNMP-CAP), we cannot select another set M'' as monitors ($M'' \cap (M \setminus M') = \emptyset$), such that $|M''| < |M'|$ and the given network is still k-identifiable using the monitors in $(M \setminus M') \cup M''$ ($M'' \neq \emptyset$ as proved in (1)). Suppose the given network is decomposed into a set of k-components ($\{\mathcal{D}_i\}_{i=1}^{\sigma}$), and the corresponding layer structure as that in SMP-CAP is constructed. Then we categorize monitors in M into different k-concatenated components as follows: Similar to SMP-CAP, for each monitor $m \in M$, following the order from layer 1 to layer σ, we find that the first k-concatenated component must contain m for achieving k-identifiability based on the existing monitors and the necessary monitor requirement rule (line 4 of SMP-CAP); this procedure is detailed in Algorithm 5. Note that in Algorithm 5, within the same layer, there may exist multiple k-concatenated components that involve the same monitor in M, for which Algorithm 5 randomly marks one k-concatenated component; see line 6. Moreover, in Algorithm 5, if a monitor $m' \in M$ has been marked as being involved in a k-concatenated component, then m' is regarded as an existing monitor in future monitor categorization; see line 7. In this way, a set of marked k-concatenated components is constructed, denoted by $\widehat{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$, by Algorithm 5. We claim that each monitor $m \in M$ must be involved in one k-concatenated component after running Algorithm 5 (i.e., $\widetilde{M} = M$ after the execution of Algorithm 5), because if some monitors $\widehat{M} \subseteq M$ cannot be involved in any k-concatenated component, then it implies that each k-concatenated component has involved a sufficient number of monitors when only $M \setminus \widehat{M}$ are monitors, and thus $M \setminus \widehat{M}$ can already achieve k-identifiability according to Lemma 20. Therefore, removing every single monitor $\widehat{m} \in \widehat{M}$ maintains k-identifiability, for which $\widehat{m}$ should have been removed by line 13 of MNMP-CAP. Since no monitors in M constructed by MNMP-CAP is removable, each monitor $m \in M$ must be involved in one k-concatenated component after running Algorithm 5. Accordingly, each monitor in M must be in one and only one of the k-concatenated components in set $\widehat{A}$; however, a k-concatenated component in set $\widehat{A}$ may involve more than one monitor in M.

Algorithm 5: Monitor Categorization

input : Set of monitors M selected by MNMP-CAP, and parameter k ($1 \leq k \leq |V|$)

output: Set of marked k-concatenated components $\widehat{A}$

1 $\widetilde{M} \leftarrow \emptyset$;
2 **for** *each* $m \in M$ **do**
3 **for** $q = 1, \ldots, \sigma$ **do**
4 **for** *each* k*-concatenated component* $\mathcal{A}$ *with size* q **do**
5 **if** $|B_{\mathcal{A}} \cup (V(\mathcal{A}) \cap \widetilde{M})| < k$ *and* $m \in V(\mathcal{A}) \setminus B_{\mathcal{A}} \setminus \widetilde{M}$ *($B_{\mathcal{A}}$: set of separation vertices in $\mathcal{A}$)* **then**
6 mark $\mathcal{A}$ as requiring monitor m from M;
7 $\widetilde{M} \leftarrow \widetilde{M} \cup m$;
8 go to line 3 to examine the next monitor in M;
9 **end**
10 **end**
11 **end**
12 **end**

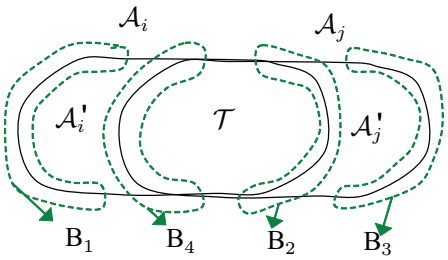


Fig. 8. Overlapped k-concatenated components.

We next consider the following question: does there exist a smaller set of monitors that satisfies the monitor requirement of any two k-concatenated components, say $\mathcal{A}_i$ and $\mathcal{A}_j$, of $\widehat{A}$? Let $M_{\mathcal{A}_i}$ and $M_{\mathcal{A}_j}$ be the sets of monitors that are marked by Algorithm 5 to be included in $\mathcal{A}_i$ and $\mathcal{A}_j$. Note that $M_{\mathcal{A}_i} \cap M_{\mathcal{A}_j} = \emptyset$ according to Algorithm 5. There are four cases for $\mathcal{A}_i$ and $\mathcal{A}_j$:

(2-i) $V(\mathcal{A}_i) \cap V(\mathcal{A}_j) = \emptyset$. In this case, since $\mathcal{A}_i$ and $\mathcal{A}_j$ do not share any common nodes, the monitor requirement for these two k-concatenated components cannot be satisfied by selecting some common nodes between $\mathcal{A}_i$ and $\mathcal{A}_j$, i.e., the number of monitors cannot be reduced. Thus, $|M_{\mathcal{A}_i}|$ and $|M_{\mathcal{A}_j}|$ monitors are required in $\mathcal{A}_i$ and $\mathcal{A}_j$.

(2-ii) $\mathcal{A}_i$ and $\mathcal{A}_j$ only have common separation vertices. Note that the monitors in $M_{\mathcal{A}_i}$ and $M_{\mathcal{A}_j}$ cannot be separation nodes in $\mathcal{A}_i$ and $\mathcal{A}_j$ according to the monitor selection rule in line 5 of SMP-CAP. Therefore, even if $\mathcal{A}_i$ and $\mathcal{A}_j$ have common separation vertices, there does not exist valid common nodes for monitor selection such that the monitor requirement for both $\mathcal{A}_i$ and $\mathcal{A}_j$ are satisfied, and the total the number of monitors can be reduced. Therefore, it is necessary to have $|M_{\mathcal{A}_i}|$ and $|M_{\mathcal{A}_j}|$ monitors in $\mathcal{A}_i$ and $\mathcal{A}_j$.

(2-iii) $\mathcal{A}_i$ and $\mathcal{A}_j$ have common k-components and one is *not* a subgraph of the other one. According to the construction rule of k-concatenated components, if $\mathcal{A}_i$ and $\mathcal{A}_j$ have overlaps other than common separation vertices, then the common vertices between $\mathcal{A}_i$ and $\mathcal{A}_j$ must form one or more k-concatenated components. As illustrated in Fig. 8, let $\mathcal{T}$ be the common k-concatenated components between $\mathcal{A}_i$ and $\mathcal{A}_j$, i.e., $\mathcal{A}_i = \mathcal{A}_i' + \mathcal{T}$ and $\mathcal{A}_j = \mathcal{A}_j' + \mathcal{T}$. Let $B_{\mathcal{A}}$ denote the set of separation vertices in k-concatenated component $\mathcal{A}$. Fig. 8 shows that $B_{\mathcal{A}_i'} = (B_1 \cup B_4)$, $B_{\mathcal{A}_j'} = (B_2 \cup B_3)$, where B_4 (B_2) is the common node set between $\mathcal{A}_i'$ and $\mathcal{A}_j$ ($\mathcal{A}_i$ and $\mathcal{A}_j'$). Then we have $B_{\mathcal{A}_i} \supseteq (B_1 \cup B_2)$, and $B_{\mathcal{A}_j} \supseteq (B_3 \cup B_4)$. Note that $B_1 \cup B_2$ ($B_3 \cup B_4$) may not include all separation vertices in $\mathcal{A}_i$ ($\mathcal{A}_j$). To reduce the total number of monitors while satisfying the monitor requirement for both $\mathcal{A}_i$ and $\mathcal{A}_j$,

the only possible way is to reselect monitors in the common k-concatenated components $\mathcal{T}$. Now we prove it is impossible to reselect a smaller set of monitors in $\mathcal{T}$: According to the monitor categorization rule in line 5 of Algorithm 5, we have $|B_1| + |B_2| + |M_{\mathcal{A}_i}| < k$ for $\mathcal{A}_i$ and $|B_3| + |B_4| + |M_{\mathcal{A}_j}| < k$ for $\mathcal{A}_j$ (as $M_{\mathcal{A}_i} \neq \emptyset$ and $M_{\mathcal{A}_j} \neq \emptyset$, the total number of separation vertices in $\mathcal{A}_i$ and $\mathcal{A}_j$ cannot be greater than or equal to k). To reselect monitors for satisfying the monitor requirement in $\mathcal{A}_i$ and $\mathcal{A}_j$, we must also satisfy the monitor requirement in $\mathcal{A}'_i$ and $\mathcal{A}'_j$. Suppose the original monitor placement $M_{\mathcal{A}_i}$ and $M_{\mathcal{A}_j}$ can satisfy the monitor requirement in $\mathcal{A}'_i$ and $\mathcal{A}'_j$, then it implies that $|B_1| + |B_4| + |M_{\mathcal{A}_i}| \geq k$ and $|B_2| + |B_3| + |M_{\mathcal{A}_j}| \geq k$, which is impossible, because $|B_1| + |B_2| + |B_3| + |B_4| + |M_{\mathcal{A}_i}| + |M_{\mathcal{A}_j}| < 2k$. Therefore, under the monitor placement $M_{\mathcal{A}_i}$ and $M_{\mathcal{A}_j}$, either $\mathcal{A}'_i$ must contain $M_{\mathcal{A}_i}$ and $\mathcal{A}_j$ must contain $M_{\mathcal{A}_j}$, or $\mathcal{A}_i$ must contain $M_{\mathcal{A}_i}$ and $\mathcal{A}'_j$ must contain $M_{\mathcal{A}_j}$. In either case, the two k-concatenated components only have common separation vertices (no common k-components). According to (2-ii), all monitors in $M_{\mathcal{A}_i}$ and $M_{\mathcal{A}_j}$ are therefore necessary to achieve k-identifiability. Thus, even if $\mathcal{A}_i$ and $\mathcal{A}_j$ have common k-components, we can replace $\mathcal{A}_i$ by $\mathcal{A}'_i$ (or $\mathcal{A}_j$ by $\mathcal{A}'_j$) without common k-components, and $|M_{\mathcal{A}_i}|$ and $|M_{\mathcal{A}_j}|$ monitors are still required in $\mathcal{A}'_i$ and $\mathcal{A}_j$ (or $\mathcal{A}_i$ and $\mathcal{A}'_j$).

(2-iv) one of $\mathcal{A}_i$ and $\mathcal{A}_j$ is a subgraph of the other one. Without loss of generality, suppose $V(\mathcal{A}_i) \subseteq V(\mathcal{A}_j)$. Then $\mathcal{A}_j$ must be in a higher layer. In this case, monitors in $M_{\mathcal{A}_i}$ are necessary in $\mathcal{A}_i$ to achieve overall k-identifiability. Then effectively, all monitors in $M_{\mathcal{A}_i}$ are also monitors in $\mathcal{A}_j$. If there are redundant monitors in $\mathcal{A}_j$, then these redundant monitors must be in $M_{\mathcal{A}_j}$, which should have been removed by line 13 of MNMP-CAP. Therefore, $|M_{\mathcal{A}_i}|$ and $|M_{\mathcal{A}_j}|$ monitors are required in $\mathcal{A}_i$ and $\mathcal{A}_j$.

In summary, in $\widehat{A}$, if two k-concatenated components fall into case (2-iii), then these two k-concatenated components can be replaced by other k-concatenated components so that the replacement components fall into case (2-ii). This updated $\widehat{A}$ can be further updated recursively until no case (2-iii) exists; let the final updated $\widehat{A}$ be $\widehat{A}'$. Then for any two k-concatenated components in $\widehat{A}'$, they either do not have common k-components (the monitor requirement in one component is independent of the monitor requirement in the other one) or one is a subgraph of the other (the monitors in the subgraph are all usable to its parent super-graph; however, the super-graph still needs more necessary monitor placement on top of the monitors already in its subgraph). In other words, each k-concatenated component in $\widehat{A}'$ *independently* requires a minimum number of nodes to be monitors even if its subgraphs may already select necessary monitors. This minimum monitor requirement can just be met by the set of monitors generated by MNMP-CAP. Therefore, MNMP-CAP only selects monitors as necessary (without which, $\mathcal{G}$ is guaranteed to be not k-identifiable) as that in SMP-CAP, and thus set M'' cannot be found, i.e., no fewer than $|M|$ monitors can achieve k-identifiability. ■

B. Auxiliary Algorithm

Auxiliary algorithm, Algorithm A, of OMP-CSP is invoked when the input network is 2-connected.

For a given 2-connected network, if it is a PLC, then two randomly chosen monitors (line 2 of Algorithm A) can ensure the identifiability of any single-node failure, which is proved in Theorem 15.

If $\mathcal{G}$ itself is not a PLC, we apply Observation 2 to find identifiable nodes by lines 4–5, and get the tailored graph $\mathcal{G}'$ containing unidentifiable nodes by line 6, after which Observation 3 is applied in line 7 to place additional monitors. If no identifiable nodes can be found, then it implies that $\mathcal{G}$ contains one and only one polygon, to which Observation 2 cannot be applied. For such a case, nevertheless, we have two scenarios where necessary monitors can be deployed: (a) First, all PLCs are non-bonds (line 9). In this scenario, there is no difference in selecting any non-agent node to be the first monitor. Hence, line 10 randomly selects a non-agent node as a monitor. Then we

apply Observation 3 to the remaining graph obtained by line 11. (b) Second, $\exists$ at least one bond PLC, denoted by Λ', in $\mathcal{G}$ (line 13). Let the two neighboring PLCs of Λ' be Λ_1 and Λ_2. Then either Λ_1 or Λ_2 should contain a monitor; otherwise, the two end-points of Λ' are not distinguishable. For example, to distinguish between v_4 and v_9 in Fig. 5, one of the neighboring PLCs should contain a monitor. In this regard, we select two monitor candidates w_1 and w_2 by lines 15–19; however, we do not know whether selections of w_1 or w_2 will generate the optimal solution. We address this issue by starting from w_i ($i = 1, 2$) being a monitor, and then apply Observations 2–3 to get the monitor placement w.r.t. w_i (lines 20–24). Then we select the placement with the minimum number of monitors in line 25 as the final output of OMP-CSP.

Example: One example is illustrated in Fig. 5. In Fig. 5, selecting nodes a, b, and c as monitors forms the optimal monitor placement for 1-identifiability, where a is selected by line 19, and b and c are selected by lines 20–24.

Complexity: In Algorithm A, all PLCs are precomputed in the master algoirithm (OMP-CSP). Then Algorithm A essentially processes all PLCs, each in constant time. Therefore, the total time complexity is $O(|T|)$ (T is the set of triconnected components) for Algorithm A.

C. Proof of Lemma 14

According to the definition, a PLC in $\mathcal{G}$ must be a subgraph of a biconnected component (say $\mathcal{B}$) in $\mathcal{G}$. To get all PLCs in $\mathcal{B}$, we can first decompose $\mathcal{B}$ into triconnected components (forming set T). Then for a triconnected component $\mathcal{T} \in T$ which is not a triangle that can be merged with other triangles to form a polygon, let graph Λ' be the maximum graph union of triconnected components in T such that (i) $V(\mathcal{T}) \subseteq V(\Lambda')$, (ii) each involved triangle is not mergeable with other triangles, and (iii) Λ' is 2-connected. Removing all virtual links in Λ' yields the corresponding PLC Λ. It has been proved in [22] that in any given network, after triconnected component decomposition, all triconnected components that are not mergeable with other triangles are unique. In other words, the decomposition uncertainty only lies in mergeable triangles. Therefore, PLC Λ ($|\Lambda| > 2$, i.e., not a bond) constructed above contains a fix set of triconnected components. In this way, all other non-bond PLCs can be found in the rest of $\mathcal{B}$. For any two non-bond PLCs constructed in this way, they can have at most one common node. This is because if they have more than one common node, then it implies the graph union above is not maximum. After constructing all these non-bond PLCs, if not all links in $\mathcal{B}$ are covered, then each remaining link together with its end-points is a bond PLC, because each of these links is part of a mergeable triangle. In summary, any biconnected component can be uniquely decomposed into bond and non-bond PLCs, and obviously the PLC decomposition in one biconnected component is independent of other biconnected components, thus completing the proof. ■

D. Proof of Theorem 15

We randomly choose two nodes in PLC Λ as monitors, denoted by m_1 and m_2. Besides m_1 and m_2, we assume that there still exists at least one non-monitor, since the monitor failure can be directly determined.

In Λ, suppose there exist two non-monitors $v_1, v_2 \in \Lambda$, where any measurement path traversing v_1 must also traverse v_2, i.e., v_1 and v_2 are not distinguishable. This implies that v_2 is not measurable using simple paths when v_1 is removed. This case is illustrated in Fig. 9, where any two nodes in $\{v_1, v_2, v_3, v_4\}$ form a 2-vertex-cut, and there is no direct link connecting any two nodes in $\{v_1, v_2, v_3, v_4\}$. In Fig. 9, there is no cycle-free path traversing one and only one of v_1 and v_2; such a network must contain a polygon (e.g., $\{v_1, v_2, v_3, v_4\}$ in Fig. 9 are four nodes in the same polygon); therefore, Λ is not a PLC, contradicting the assumption.

Algorithm A: *Monitors-in-Biconneted-Network*

input : 2-connected network $\mathcal{G}$ and its PLCs

output: Set of monitors that achieves 1-identifiability in $\mathcal{G}$ under CSP

if $\mathcal{G}$ *is a PLC* **then**
  randomly select two nodes as monitors; return;
end
if $\exists$ *non-empty set* A *containing all PLCs with 4 or more neighboring PLCs within* $\mathcal{G}$ **then**
  find set C containing all neighboring PLCs of each PLC in set A within $\mathcal{G}$;
  $\mathcal{G}' \leftarrow \mathcal{G} \ominus (A \cup C)$;
  Monitors-in-Polygon-less-Network($\mathcal{G}'$, S_a), where S_a denotes the set of agents in $\mathcal{G}$;
`//In the following cases, G must contain only one polygon;`
else if *all PLCs in* $\mathcal{G}$ *are non-bonds* **then**
  randomly select a non-agent node in a PLC, denoted by Λ, as a monitor;
  $\mathcal{G}' \leftarrow \mathcal{G} \ominus (\{\Lambda\} \cup E)$, where E is the set containing all neighboring PLCs of Λ;
  Monitors-in-Polygon-less-Network($\mathcal{G}'$, S_a);
else `//∃ at least one bond PLC`
  randomly select a bond PLC Λ' with two end-points v_1 and v_2 and two neighboring PLCs Λ_1 ($v_1 \in \Lambda_1$) and Λ_2 ($v_2 \in \Lambda_2$);
  if Λ_1 *(Λ_2) is a bond* **then**
    $w_1 \leftarrow v_1$ ($w_2 \leftarrow v_2$);
  else
    w_1 (w_2) $\leftarrow$ a (random) non-agent node in Λ_1 (Λ_2);
  end
  foreach $i = 1, 2$ **do**
    select w_i as a monitor;
    $\mathcal{G}'_i \leftarrow \mathcal{G} \ominus \Gamma_{w_i}$, where Γ_{w_i} is the set involving (i) PLCs that contain w_i, and (ii) neighboring PLCs of the PLCs in (i) if w_i is not an agent;
    Monitors-in-Polygon-less-Network($\mathcal{G}'_i$, S_a);
  end
  in above two monitor placements, select the one with the minimum number of monitors as the final output;
end

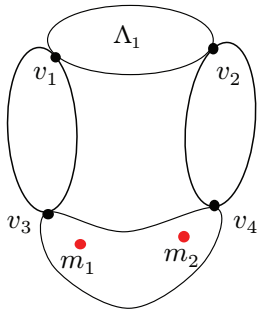


Fig. 9. Undistinguishable nodes v_1 and v_2 in PLC Λ_1.

Suppose two non-monitors v with $v \in \Lambda$ and w with $w \notin \Lambda$ are not distinguishable. However, there exists a simple path traversing v using only the nodes in Λ. Thus, v and w are distinguishable, contradicting the assumption.

Therefore, if a PLC contains at least two monitors, then all its non-monitors are identifiable. ■

E. Proof of Corollary 16

(1) Consider PLC Λ with two and only two cut-vertices. We know that these two cut-vertices must have two external vertex-disjoint paths to different monitors. Using these two external monitor connections as two segments in the measurement path, we can follow the same argument in Appendix D to prove that any two non-cut-vertices in Λ are distinguishable. Moreover, for any non-cut-vertex v in Λ, we can always find a measurement path traversing v (or not traversing v) since Λ is 2-connected. Therefore, v is also distinguishable from any node outside Λ. Hence, if Λ contains two cut-vertices, then all its non-cut-vertices are identifiable.

(2) Consider PLC Λ with three or more cut-vertices. In this case, we know from (1) that any non-cut-vertex is 1-identifiable and all cut-vertices have external vertex-disjoint paths to different monitors. Thus, it suffices to show the cut-vertices on Λ are also identifiable. In this case, for any cut-vertex v_c, we can construct a path in Λ without traversing v_c, since the removal of v_c does not separate Λ (Λ is 2-connected). Therefore, cut-vertex v_c is distinguishable from any other single node failure in the entire network. Hence, if Λ contains three or more cut-vertices, then all its nodes are identifiable. ■