Robust Monitor Placement for Network Tomography in Dynamic Networks

Ting He*, Liang Ma*, Athanasios Gkelias†, Kin K. Leung†, Ananthram Swami‡, and Don Towsley§
*IBM T. J. Watson Research Center, Yorktown, NY, USA. Email: {the, maliang}@us.ibm.com
†Imperial College, London, UK. Email: {a.gkelias, kin.leung}@imperial.ac.uk
‡Army Research Laboratory, Adelphi, MD, USA. Email: ananthram.swami.civ@mail.mil
§University of Massachusetts, Amherst, MA, USA. Email: towsley@cs.umass.edu

***Abstract*—We consider the problem of placing the minimum number of monitors in a communication network with possible topology changes to identify additive link metrics from path metrics. The core of our solution is a suite of robust monitor placement algorithms with different performance-complexity tradeoffs that guarantee network identifiability for the multiple possible topologies. In particular, we show that the optimal (i.e., minimum) monitor placement is the solution to a generalized hitting set problem, where we provide a polynomial-time algorithm to construct the input. Although the optimal placement is NP-hard in general, we identify non-trivial special cases that can be solved efficiently. We further demonstrate how the proposed algorithms can be augmented to handle unpredictable topology changes and tradeoffs between monitor cost and adaptation cost. Our evaluations on mobility-induced dynamic topologies verify the effectiveness and robustness of the proposed algorithms.**

I. Introduction

Network tomography refers to the methodology of inferring internal performance metrics (e.g., link delays/losses) of a network from external measurements between nodes with monitoring capabilities, referred to as *monitors*. Since its introduction [1], network tomography has attracted significant interest in the research community as a promising alternative to the approach of direct measurement. Traditionally, network monitoring systems rely on diagnostic tools such as *traceroute*, *pathchar* [2], and *Network Characterization Service (NCS)* [3] to directly measure the performance of individual links via active probes, but these techniques suffer a high measurement overhead and require cooperation from every internal node. In contrast, tomography-based monitoring only requires cooperation of nodes designated as monitors and can utilize performance experienced by data packets to reduce the need of active probes [4].

A major challenge in applying network tomography is the lack of uniqueness in the inferred link metrics. For example, consider the inference of link metrics that are *additive* (e.g., delays, jitters, log of packet delivery ratio). Network tomography infers such link metrics by solving a system of linear equations, where the unknown variables are the link metrics, and each measurement path provides an equation that relates the metrics of traversed links to the end-to-end measurement on this path. From linear algebra, we know that the system has a unique solution if and only if the measurement paths span the entire link space, i.e., the number of *linearly independent* paths equals the number of links. However, past experience shows that without careful design, it is frequently impossible to uniquely determine all link metrics from path measurements [5], [6], [7].

This problem, known as the *identifiability* problem, has been recognized in the literature, with several solutions proposed to place monitors so as to identify all link metrics under the assumption that the network topology is fixed [8], [9]. While the fixed-topology assumption is valid in wired networks, applying network tomography to wireless networks faces the additional challenge that the network topology may vary dynamically at runtime due to factors such as node mobility, node status (up/down), and channel variation. While a straightforward solution is to handle the changes *reactively* by computing a new monitor placement after each topology change, such a solution can lead to frequent reconfigurations and instability in the monitoring system. To maintain seamless monitoring of link performance in such dynamic networks, it is desirable to have a monitor placement strategy that can handle topology changes *proactively*.

In this paper, we aim to develop monitor placement algorithms that are *robust* to topology changes. In many networks, it is possible to predict topology changes with certain accuracy. For example, we can predict topology changes based on models of node mobility [10], [11], patterns of link failures [12], or frequently occurring topologies in the past. Given such prediction capabilities, we are interested in two closely-related problems: (i) During network planning, how can we place monitors to ensure identifiability under predictable topology changes? (ii) At runtime, how can we adapt the monitor placement so as to maintain identifiability under unpredictable topology changes? In both problems, we wish to use as few monitors as possible to minimize cost.

A. Related Work

Based on how measurements are performed, existing solutions on monitor placement can be categorized as: (1) placement of monitors performing round-trip probing, (2) placement of monitors performing end-to-end probing.

In category (1), each monitor (aka beacon) independently computes metrics of a subset of links by sending round-trip probes to all possible destinations along its routing tree. The

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 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.

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

goal is thus to place a minimum number of monitors whose routing trees cover all the links. The problem is proved to be NP-hard, and algorithms based on set covering are proposed to select a sufficient set of monitors. Variations have also been proposed to cover all active links under a limited number of link failures or route changes [13], [14].

In category (2), monitors jointly infer link metrics from end-to-end measurements between themselves. Under the assumption that monitors can measure arbitrary cycles or paths (possibly) containing cycles, [8] derives the first necessary and sufficient condition on the network topology for identifying additive link metrics from end-to-end measurements. The condition is later modified by [9] to incorporate an additional constraint that measurement paths must be cycle-free, which allows the use of data packets for measurements. Based on the modified condition, [9] develops an algorithm to place a minimum number of monitors to identify all link metrics. All existing works assume a fixed network topology. We adopt the measurement model in [9] but consider arbitrary topology changes.

A key enabler for robust monitor placement is the knowledge of topologies after changes. Topology prediction has been studied in the context of wireless ad-hoc and vehicular networks, where techniques including adaptive filtering and fluid dynamic modeling have been proposed to predict link changes [10], [11]. In this work, we assume the existence of such a topology predictor and focus on developing robust monitor placement algorithms based on the predicted topologies.

B. Summary of Contributions

We study robust monitor placement for inferring additive link metrics from end-to-end measurements along cycle-free paths under dynamic topology changes. Our contributions are:

1) We develop robust monitor placement algorithms that place monitors to *simultaneously* achieve identifiability for a given set of topologies, including: (i) a one-shot placement algorithm that applies an existing algorithm for static networks to an aggregate topology, (ii) an incremental placement algorithm that sequentially places monitors in each topology, and (iii) an optimal placement algorithm that jointly considers monitor requirements of different topologies by casting the problem as a generalized hitting set problem. All these algorithms guarantee identifiability with different tradeoffs between the number of monitors and the complexity. We also provide an algorithm to identify and remove unnecessary monitors by solving the dual of the joint placement problem.

2) We show that the optimal monitor placement is NP-hard but can be approximated to a logarithmic factor by a greedy heuristic. We further identify several cases where the problem can be solved optimally. In particular, one case corresponds to static networks with fixed topologies, explaining why the problem is solvable (by [9]) in this case.

3) We present several extensions to the above solution to address practical issues including topology selection, unpredictable changes, and bounded number of monitors.

4) We evaluate the proposed solutions on realistic dynamic topologies driven by node mobility traces. Besides verifying the performance-complexity tradeoffs of the algorithms, our results show that it is possible for a small percentage of monitors ($10-30\%$) to maintain identifiability in the presence of hundreds of topology changes, and our monitor placement is highly robust against errors in topology prediction.

The rest of the paper is organized as follows. Section II formulates the problem. Section III reviews existing results. Section IV presents new solutions, for which Section V identifies optimality conditions. Section VI discusses extensions to handle practical challenges. Section VII evaluates the proposed solutions. Then Section VIII concludes the paper. All proofs are available at [15].

II. Problem Formulation

A. Network Models

Consider a fixed set of nodes V which form a network with time-varying topologies. We assume that the network topology can be predicted (up to a certain accuracy) using existing topology prediction models such as [10], [11]. Based on the prediction, we identify a set of topologies of interest, represented by undirected graphs $\{\mathcal{G}_t : t = 1, \ldots, T\}$, where $\mathcal{G}_t = (V, L_t)$ is a graph representing the t-th topology[1]. We note that these topologies do not need to occur sequentially in time and do not need to be exhaustive. We defer the selection of topologies to Section VI-A. Although we focus on link changes in this paper, node changes can also be handled by modeling them as special link changes; see Section VI-D.

B. Objective of Network Tomography

Given a topology $\mathcal{G} = (V, L)$ (subscript omitted), network tomography aims at inferring the performance metrics of individual links in L from end-to-end metrics along a set of measurement paths P. In particular, we are interested in inferring *additive metrics* where the path metric equals summation of corresponding link metrics; examples of such metrics include delay, jitter, and log delivery rate (under the assumption of independent losses across links). A basic requirement of network tomography is the uniqueness of the solution.

Definition 1. A network $\mathcal{G}$ is *identifiable* if all its link metrics can be uniquely determined from path metrics.

From linear algebra, we know that $\mathcal{G}$ is identifiable if and only if the rank of the measurement paths equals the number of links. We assume that we can only monitor paths that start/end at certain nodes designated as *monitors*. We further assume that monitors can control the routing of measurement packets as long as the path starts and ends at different monitors and does not contain repeated nodes, i.e., it is a *simple path* between monitors. Under this assumption, the identifiability of $\mathcal{G}$ is completely determined by the placement of monitors. If $\mathcal{G}$ is identifiable under a given monitor placement, we say that this placement *identifies* $\mathcal{G}$. A partitioned network is identifiable if and only if all its connected components are identifiable. In particular, we assume that all isolated nodes must be monitors for the network to be identifiable.

[1]Our monitor placement algorithms can handle arbitrary topologies, although their performance depends on the kind of topologies; see Section V.

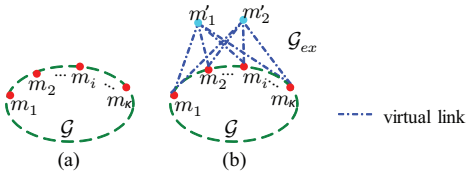


Fig. 1. (a) $\mathcal{G}$ with κ ($\kappa \geq 3$) monitors; (b) $\mathcal{G}_{ex}$ with two virtual monitors.

C. Objective of Monitor Placement

Our main objective is to select a smallest subset of nodes as monitors to identify all the topologies of interest. We will discuss later (Section VI) how such a monitor placement can be combined with temporary monitors to maintain identifiability in case of unpredictable topology changes.

III. Monitor Placement for Static Networks

We start by reviewing existing results from [9] for a static network with a fixed topology. We then extract insights from these results to motivate our solutions for dynamic networks.

A. Identifiability Condition

Definition 1 does not allow efficient testing of identifiability as there are exponentially many measurement paths. Existing work [9] has established an equivalent condition in terms of an *extended graph* $\mathcal{G}_{ex}$ constructed as follows: given a network $\mathcal{G}$ with κ ($\kappa \geq 3$) monitors[2], $\mathcal{G}_{ex}$ is obtained by adding two *virtual monitors* m'_1 and m'_2, and 2κ *virtual links* between each pair of virtual-actual monitors, as illustrated in Fig. 1. The identifiability of $\mathcal{G}$ is characterized by the following condition.

Theorem III.1. [9] Given κ ($\kappa \geq 3$) monitors, $\mathcal{G}$ is identifiable *if and only if* its extended graph $\mathcal{G}_{ex}$ is 3-vertex-connected, i.e., it remains connected after removing any two nodes.

B. Minimum Monitor Placement for Static Networks

Based on the identifiability condition in Theorem III.1, [9] gives an algorithm, called *Minimum Monitor Placement (MMP)*, that places a minimum set of monitors to ensure identifiability. We briefly review MMP for completeness and refer to [9] for details. MMP works by decomposing $\mathcal{G}$ into subgraphs with certain properties defined as follows.

Definition 2. A *k-connected component* of $\mathcal{G}$ is a maximal sub-graph of $\mathcal{G}$ that is either (i) k-vertex-connected, or (ii) a clique with up to k vertices. The case of $k = 2$ is called a *biconnected component*, and $k = 3$ a *triconnected component*.

Biconnected components are separated by cut-vertices, and triconnected components are separated by cut-vertices and 2-vertex cuts[3]. Common vertices between a given component and its neighboring components are called *separation vertices*.

Algorithm 1 summarizes the steps of MMP. Given a sub-graph $\mathcal{D}$, let $V(\mathcal{D})$ denote all the nodes in $\mathcal{D}$, $S_{\mathcal{D}}$ the separation vertices in $\mathcal{D}$, and $M_{\mathcal{D}}$ the monitors placed at internal nodes (i.e., non-separation vertices) of $\mathcal{D}$. For each connected component, MMP first places monitors at nodes with degree less than three (line 2). It then decomposes the component into biconnected and then triconnected components[4] (lines 3 and 5). Based on the decomposition, MMP selects monitors such that each bi/triconnected component with at least three nodes has at least three nodes that are separation vertices or monitors (lines 8 and 10). Finally, it ensures that each connected component with at least three nodes has at least three monitors[5] (line 12).

Algorithm 1: Minimum Monitor Placement (MMP) [9]

```
   input : Network topology G
   output: A subset of nodes in G as monitors
1  foreach connected component C_k of G do
2      choose all the nodes with degree less than 3 as monitors;
3      partition C_k into biconnected components B_1, B_2, ...;
4      foreach biconnected component B_i with at least 3 nodes do
5          partition B_i into triconnected components T_1, T_2, ...;
6          foreach triconnected component T_j do
7              if |S_{T_j}| + |M_{T_j}| < 3 then
8                  randomly place 3 − |S_{T_j}| − |M_{T_j}| monitors at
                   nodes in V(T_j) \ (S_{T_j} ∪ M_{T_j});
9          if |S_{B_i}| + |M_{B_i}| < 3 then
10             randomly place 3 − |S_{B_i}| − |M_{B_i}| monitors at
               nodes in V(B_i) \ (S_{B_i} ∪ M_{B_i});
11     if |M_{C_k}| < 3 then
12         randomly place min(3, |V(C_k)|) − |M_{C_k}| monitors at
           nodes in V(C_k) \ M_{C_k};
```

MMP is efficient with a complexity of $O(|V| + |L|)$ for $\mathcal{G} = (V, L)$ [9]. It is also provably optimal as stated below.

Theorem III.2. [9] Given an arbitrary network topology $\mathcal{G}$, MMP places the minimum number of monitors to identify $\mathcal{G}$.

Key observations: MMP places two types of monitors:

- deterministically placed monitors: nodes with degree less than three have to be monitors (line 2), as otherwise their neighboring links are not measurable by simple paths;
- randomly placed monitors: MMP ensures at least three nodes that are either separation vertices or monitors in each bi/tri/1-connected component, but the exact monitor locations can be arbitrary (lines 8, 10, 12).

We will leverage these observations in deriving monitor placement algorithms for dynamic networks.

IV. Monitor Placement for Dynamic Networks

A dynamic network may have multiple topologies during its lifetime, represented by $\{\mathcal{G}_t : t = 1, \ldots, T\}$. Based on the identifiability condition in Theorem III.1, the problem of robust monitor placement can be cast as follows: select a minimum set of nodes $M \subseteq V$ as monitors such that for each $t = 1, \ldots, T$, the extended graph $\mathcal{G}_{t,ex}$ constructed from topology $\mathcal{G}_t$ and monitors M is 3-vertex-connected.

Given the results for static networks (Section III), one can guarantee identifiability for all the T topologies by applying MMP to compute a monitor placement M_t for each $\mathcal{G}_t$, and then taking the union $M = \bigcup_{t=1}^{T} M_t$. Such a solution, however, may select redundant monitors. For example, to identify the topologies $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 2, MMP may select monitors $M_1 = \{a, b, c\}$ for $\mathcal{G}_1$, and $M_2 = \{c, f, h\}$ for $\mathcal{G}_2$,

[2]It has been shown [9] that a network with more than one link needs at least three monitors to identify all link metrics.

[3]Since not all subgraphs separated by 2-vertex cuts form triconnected components according to Definition 2, an iterative procedure is proposed in [9] to fix this issue by adding virtual links between nodes in 2-vertex cuts.

[4]There exist fast graph algorithms for such decomposition; see [9].

[5]In a connected component with one or two nodes, all nodes are monitors.

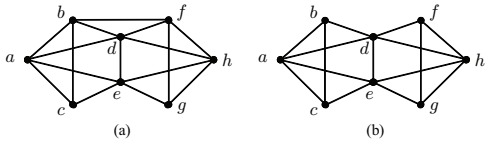


Fig. 2. Monitor placement for two topologies (the optimal monitor placement is $\{c, f, h\}$): (a) $\mathcal{G}_1$ (3-vertex-connected); (b) $\mathcal{G}_2$ (2-vertex-connected with a 2-vertex cut $\{d, e\}$).

with a total of 5 monitors. However, by Theorem III.1, M_2 already identifies both $\mathcal{G}_1$ and $\mathcal{G}_2$, i.e., two redundant monitors are selected by MMP. This example illustrates the need to jointly consider all topologies for placing monitors. In this regard, we outline a set of solutions with different performance (measured by number of monitors) and complexity.

A. One-Shot Placement

This solution is inspired by an interesting property of identifiable networks. Due to Theorem III.1, a network $\mathcal{G}$ that is identifiable under a given monitor placement M remains identifiable after adding links, because a 3-vertex-connected graph $\mathcal{G}_{ex}$ remains 3-vertex-connected after adding links. Therefore, to identify topologies $\{\mathcal{G}_t = (V, L_t) : t = 1, \ldots, T\}$, it suffices to place monitors to identify a single topology with a smaller set of links. One such topology, referred to as the *base graph*, is defined as $\mathcal{G}_b := (V, \bigcap_{t=1}^{T} L_t)$, i.e., the maximum common subgraph of $\mathcal{G}_1, \ldots, \mathcal{G}_T$. By definition, each $\mathcal{G}_t$ can be generated from $\mathcal{G}_b$ by adding links, and thus the above observation leads to the following result.

Lemma IV.1. If a monitor placement M identifies the base graph $\mathcal{G}_b$ of $\{\mathcal{G}_t : t = 1, \ldots, T\}$, then it identifies each $\mathcal{G}_t$.

This result motivates a one-shot placement algorithm: first, compute the base graph $\mathcal{G}_b$ by identifying common links in all topologies, and then apply MMP to compute a monitor placement that identifies $\mathcal{G}_b$. Generally, one-shot placement uses more than the minimum number of monitors; however, it is optimal under certain conditions on the input topologies (see Section V-A).

Complexity: The base graph can be computed in $O(T \min_t |L_t|)$ time (assuming the existence of a link in a topology can be checked in constant time), and applying MMP to the base graph takes $O(|V| + \min_t |L_t|)$ time. The overall complexity of the one-shot placement algorithm is therefore $O(|V| + T \min_t |L_t|)$.

B. Incremental Placement

A natural alternative to the one-shot placement is to apply MMP to each topology. Instead of applying the original MMP, however, we use a variation that takes into account existing monitors to reduce the number of additional monitors.

Specifically, in addition to a topology $\mathcal{G}$, we modify MMP to take an additional input of M_0 that denotes the set of existing monitors. Redefine $M_{\mathcal{D}}$ to denote the internal (i.e., non-separation-vertex) monitors in a subgraph $\mathcal{D}$ among both existing and newly placed monitors. Then the modified algorithm, referred to as *incremental MMP (IMMP)*, follows the same steps as in Algorithm 1, except that it only returns the set of newly placed monitors M_a. We have the following property of IMMP (see [15] for the proof of this and other theorems.).

Lemma IV.2. Given a topology $\mathcal{G}$ and existing monitors M_0, IMMP places the minimum number of additional monitors M_a such that $M_0 \cup M_a$ identifies $\mathcal{G}$.

The incremental placement algorithm based on IMMP works as follows: for each $\mathcal{G}_t = \mathcal{G}_1, \ldots, \mathcal{G}_T$ ($M_0 = \emptyset$),

1) $M_{a,t} \leftarrow$IMMP($\mathcal{G}_t$, M_{t-1});
2) $M_t \leftarrow M_{t-1} \cup M_{a,t}$.

Then M_T is guaranteed to identify $\mathcal{G}_1, \ldots, \mathcal{G}_T$.

Complexity: Since IMMP has the same complexity as MMP, i.e., $O(|V| + |L_t|)$ for each $\mathcal{G}_t$, the overall complexity of the incremental placement algorithm is $O(T|V| + \sum_{t=1}^{T} |L_t|)$.

C. Joint Placement

Despite being locally optimal (Lemma IV.2), the overall placement generated by IMMP is suboptimal in general due to the lack of a *joint* consideration of monitor requirements of all topologies. Jointly considering all topologies is highly nontrivial, as each topology can have exponentially many, equally optimal monitor placements, corresponding to all combinations of possible outcomes of lines 8, 10, and 12 of Algorithm 1. The brute-force strategy of enumerating all possible placements for individual topologies to find a minimum union is clearly inefficient. Our idea in addressing this issue is to decouple the problem into two stages: (i) constraint characterization and (ii) monitor selection.

1) Constraints on Monitor Placement: Our key insight is that all possible placements generated by MMP can be succinctly encoded into a set of constraints. Specifically, instead of randomly picking one placement as in MMP, we record the constraints on monitor placement in the form of set-integer pairs $\mathcal{F} = \{(S_i, k_i) : i = 1, 2, \ldots\}$, where each $S_i \subseteq V$ is a set of candidate monitors, and k_i is the minimum number of monitors selected from this set. For example, if MMP places a monitor at a randomly selected node in S, we can represent all possible placements by a constraint $(S, 1)$.

Given a topology $\mathcal{G}$, these constraints can be computed efficiently using an algorithm called *Feasible Monitor Placement (FMP)*, shown in Algorithm 2. FMP follows a procedure similar to MMP, except that instead of selecting specific nodes as monitors, FMP records the constraints on where monitors should be placed such that there is one constraint for each node that must be a monitor (line 3) and each component that requires at least one monitor (lines 9, 11, 12). Here $S_{\mathcal{D}}$ again denotes the set of separation vertices in subgraph $\mathcal{D}$. We can show that the resulting constraints are both sufficient and necessary for identifying $\mathcal{G}$.

Lemma IV.3. Any monitor placement $M \subseteq V$ identifies $\mathcal{G}$ if and only if M satisfies the constraints $\mathcal{F}$ computed by FMP, i.e., $|M \cap S_i| \geq k_i$ for all $(S_i, k_i) \in \mathcal{F}$.

Discussion: It is possible that not all constraints are needed, e.g., if the total number of monitors required by the triconnected components within a biconnected component exceeds the number of monitors required by the biconnected component, then we do not need a separate constraint for this biconnected component. We can avoid redundant constraints by counting the number of monitors in each bi/tri/1-connected component according to existing constraints (by mimicking

Algorithm 2: Feasible Monitor Placement (FMP)

input : Network topology $\mathcal{G}$
output: Monitor placement constraints $\mathcal{F}$
1 **foreach** *connected component* $\mathcal{C}_k$ *of* $\mathcal{G}$ **do**
2 **foreach** *node* v *with degree less than* 3 **do**
3 add $(\{v\},\ 1)$ to $\mathcal{F}$;
4 partition $\mathcal{C}_k$ into biconnected components $\mathcal{B}_1, \mathcal{B}_2, \ldots$;
5 **foreach** *biconnected component* $\mathcal{B}_i$ *with at least* 3 *nodes* **do**
6 partition $\mathcal{B}_i$ into triconnected components $\mathcal{T}_1, \mathcal{T}_2, \ldots$;
7 **foreach** *triconnected component* $\mathcal{T}_j$ **do**
8 **if** $|S_{\mathcal{T}_j}| < 3$ **then**
9 add $(V(\mathcal{T}_j) \setminus S_{\mathcal{T}_j},\ 3 - |S_{\mathcal{T}_j}|)$ to $\mathcal{F}$;
10 **if** $|S_{\mathcal{B}_i}| < 3$ **then**
11 add $(V(\mathcal{B}_i) \setminus S_{\mathcal{B}_i},\ 3 - |S_{\mathcal{B}_i}|)$ to $\mathcal{F}$;
12 add $(V(\mathcal{C}_k),\ \min(3, |V(\mathcal{C}_k)|))$ to $\mathcal{F}$;

MMP) and adding a new constraint to $\mathcal{F}$ only if the current number of monitors is not sufficient.

2) Constrained Monitor Selection: Given the constraints $\mathcal{F}$ obtained by applying FMP to each topology $\mathcal{G}_t$ $(t = 1, \ldots, T)$, the monitor placement problem is converted into a problem of selecting a minimum subset $M \subseteq V$ such that $|M \cap S_i| \geq k_i$ for all $(S_i,\ k_i) \in \mathcal{F}$. We refer to this problem as the *minimum hitting set problem (min-HSP)* with input $(V,\ \mathcal{F})$. In the special case of $k_i \equiv 1$, it becomes the classic hitting set problem (HSP). Because the constraints computed by FMP are necessary and sufficient for achieving identifiability (Lemma IV.3), we can obtain an optimal monitor placement by solving the corresponding min-HSP optimally.

Theorem IV.4. Let $\mathcal{F}$ be the overall set of constraints computed by applying FMP to each topology $\mathcal{G}_t$ $(t = 1, \ldots, T)$. Then the optimal solution to min-HSP$(V, \mathcal{F})$ yields an optimal (i.e., minimum) monitor placement for identifying $\mathcal{G}_1, \ldots, \mathcal{G}_T$.

The challenge is that min-HSP is NP-hard since HSP is NP-hard. Although given constraints $\mathcal{F}$, the problem of constrained monitor selection is a special case of min-HSP, we show that it is still NP-hard by a reduction from HSP.

Theorem IV.5. The optimal monitor placement for arbitrary topologies $\mathcal{G}_t$ $(t = 1, \ldots, T)$ is NP-hard.

Greedy approximation: In our problem, we can speed up computation by first filtering out nodes that must be monitors (i.e., nodes with degree less than three in at least one topology) and then applying the greedy heuristic. For our problem, the greedy heuristic works as follows: while there are unsatisfied constraints, select the monitor that helps in satisfying the maximum number of unsatisfied constraints, i.e., given the current monitors M, select v as the next monitor such that v is in the maximum number of sets among $\{S_i : (S_i,\ k_i) \in \mathcal{F},\ |S_i \cap M| < k_i\}$. It can be shown that min-HSP is a special case of integer linear programming with non-negative data [16], and the greedy heuristic achieves an approximation ratio of[6] $(1 + \log |\mathcal{F}|)$.

Complexity: Together, FMP and the greedy heuristic for min-HSP provide a joint monitor placement algorithm. FMP has the same complexity as MMP, which is $O(|V| + |L_t|)$ for each $\mathcal{G}_t$ $(t = 1, \ldots, T)$ [9]. The greedy heuristic has complexity $O(T|V|^2)$, as there are $O(T|V|)$ constraints and an $O(|V|)$-complexity update upon satisfying each constraint (to compute the number of unsatisfied constraints each candidate monitor is involved in). Thus, the overall complexity is $O(T|V|^2)$.

[6]Specifically, let $\mathcal{F}_v := \{(S_i,\ k_i) \in \mathcal{F} :\ v \in S_i\}$. Theorem 3.1 in [16] implies that the ratio between the number of monitors selected by the greedy heuristic and the minimum number of monitors is at most $1 + \max_{v \in V} \log |\mathcal{F}_v|$.

Algorithm 3: Redundant Monitor Discovery (RMD)

input : Network topology $\mathcal{G}$, monitor placement M that identifies $\mathcal{G}$
output: Monitor removal constraints $\mathcal{R}$
1 **foreach** *connected component* $\mathcal{C}_k$ *of* $\mathcal{G}$ **do**
2 **foreach** *node* v *with degree less than* 3 **do**
3 add $(\{v\},\ 0)$ to $\mathcal{R}$;
4 partition $\mathcal{C}_k$ into biconnected components $\mathcal{B}_1, \mathcal{B}_2, \ldots$;
5 **foreach** *biconnected component* $\mathcal{B}_i$ *with at least* 3 *nodes* **do**
6 partition $\mathcal{B}_i$ into triconnected components $\mathcal{T}_1, \mathcal{T}_2, \ldots$;
7 **foreach** *triconnected component* $\mathcal{T}_j$ **do**
8 **if** $|S_{\mathcal{T}_j}| < 3$ **then**
9 add $(M_{\mathcal{T}_j},\ |M_{\mathcal{T}_j}| - 3 + |S_{\mathcal{T}_j}|)$ to $\mathcal{R}$;
10 **if** $|S_{\mathcal{B}_i}| < 3$ **then**
11 add $(M_{\mathcal{B}_i},\ |M_{\mathcal{B}_i}| - 3 + |S_{\mathcal{B}_i}|)$ to $\mathcal{R}$;
12 add $(M_{\mathcal{C}_k},\ |M_{\mathcal{C}_k}| - \min(3, |V(\mathcal{C}_k)|))$ to $\mathcal{R}$;

D. Refinement of Placement

Due to the hardness of the optimal solution, the computed monitor placement generally contains more than the minimum number of monitors. A natural question is therefore how to refine this placement by removing redundant monitors without losing identifiability. As shown below, the problem of (redundant) monitor removal can also be decoupled into two stages.

1) Constraints on Monitor Removal: The problem of characterizing constraints on monitor removal is similar to that of characterizing constraints on monitor selection (Section IV-C1), and thus can be solved by following similar steps.

The algorithm, referred to as *Redundant Monitor Discovery (RMD)*, is summarized in Algorithm 3, where $M_{\mathcal{D}}$ denotes the set of internal monitors in a subgraph $\mathcal{D}$. Note that in contrast to MMP where $M_{\mathcal{D}}$ varies during monitor placement, here $M_{\mathcal{D}}$ is based on the given monitor placement and thus fixed. RMD is analogous to FMP, except that it computes constraints on monitor removal (lines 3, 9, 11, 12). Each constraint is also represented by a set-integer pair $(S_i,\ k_i)$, which means that no more than k_i monitors from S_i can be removed.

It is easy to verify that when removing monitors from a monitor placement M that identifies $\mathcal{G}$, the remaining monitors satisfy the constraints computed by FMP if and only if the removed monitors satisfy the constraints computed by RMD. This duality immediately yields the following result.

Lemma IV.6. Given a monitor placement M that identifies $\mathcal{G}$, $M \setminus M'$ identifies $\mathcal{G}$ for any $M' \subset M$ if and only if M' satisfies $\mathcal{R}$ computed by RMD, i.e., $|M' \cap S_i| \leq k_i$ for all $(S_i,\ k_i) \in \mathcal{R}$.

Discussion: As in FMP, it is also possible for RMD to generate redundant constraints. We can avoid this by removing redundant constraints before adding each new constraint, i.e., when adding a constraint $(S,\ k)$, all existing constraints $(S',\ k')$ with $S' \subseteq S$ and $k' \geq k$ can be removed.

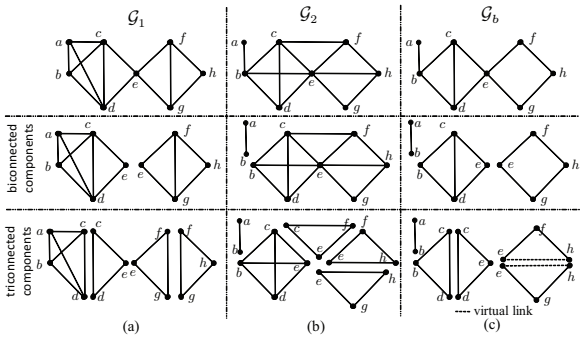


Fig. 3. Example of placement algorithms (an optimal monitor placement is $\{a, d, g, h\}$): (a) $\mathcal{G}_1$ and its decomposition; (b) $\mathcal{G}_2$ and its decomposition; (c) base graph $\mathcal{G}_b$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ and the decomposition of $\mathcal{G}_b$.

2) Constrained Monitor Removal: Given an initial placement M_0 that identifies all $\mathcal{G}_t$ $(t = 1, \ldots, T)$ and the constraints $\mathcal{R}$ computed by applying RMD to each $\mathcal{G}_t$ and M_0, the problem of removing redundant monitors in M_0 becomes a problem of selecting a maximum subset $M' \subset M_0$ such that $|M' \cap S_i| \leq k_i$ for all $(S_i,\, k_i) \in \mathcal{R}$. We refer to this problem as the *maximum hitting set problem (max-HSP)* with input $(M_0, \mathcal{R})$. The duality between the minimum monitor selection and the maximum redundant monitor removal implies an alternative way of computing an optimal placement as follows.

Corollary IV.7. Let $\mathcal{R}$ be the overall set of constraints computed by applying RMD to each of $\mathcal{G}_1, \ldots, \mathcal{G}_T$ with an initial placement $M_0 = V$, and M' be the optimal solution to max-HSP$(V, \mathcal{R})$. Then $V \backslash M'$ is an optimal monitor placement for identifying $\mathcal{G}_1, \ldots, \mathcal{G}_T$.

Unfortunately, max-HSP is again NP-hard. In fact, it is exactly as hard as min-HSP because solving a min-HSP for input $(V,\, \{(S_i,\, k_i) :\, i = 1, 2, \ldots\})$ is equivalent to solving a max-HSP for input $(V,\, \{(S_i,\, |S_i| - k_i) :\, i = 1, 2, \ldots\})$.

Greedy approximation: We can apply a greedy heuristic similar to the one used in Section IV-C2: while there are redundant monitors (i.e., monitors such that removing any one of them does not violate any constraint), remove the redundant monitor that is involved in the minimum number of constraints. In fact, it can be shown that the greedy heuristic achieves an approximation ratio of[7] $1/|\mathcal{R}|$.

Complexity: Although max-HSP has the same complexity as min-HSP on the same input, the actual complexity of monitor removal depends on the size of the initial placement $|M_0|$, which can be much smaller than $|V|$. Specifically, RMD has the same complexity as FMP, i.e., $O(T|V| + \sum_t |L_t|)$ for processing $\mathcal{G}_1, \ldots, \mathcal{G}_T$. The greedy heuristic for max-HSP has complexity $O(T|V| \cdot |M_0|)$ to select from a size-$|M_0|$ set under $O(T|V|)$ constraints. The overall complexity of RMD and greedy max-HSP is therefore $O(T|V| \cdot |M_0| + \sum_t |L_t|)$.

Example: Consider a network with the topologies $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 3. The one-shot placement first obtains the base graph $\mathcal{G}_b$ and then applies MMP to $\mathcal{G}_b$, which may select monitors $M^{\text{one}} = \{a, d, f, h, g\}$. Note that MMP is a random algorithm; in this case, it may select c instead of d. The incremental placement first applies MMP to $\mathcal{G}_1$, which may select $\{b, c, f, h\}$ as monitors, and then applies IMMP to $\mathcal{G}_2$, which may select additional monitors $\{a, g\}$, yielding $M^{\text{inc}} = \{b, c, f, h, a, g\}$. The joint placement first computes the constraints: $\mathcal{F}_1 = \{(\{a, b\}, 1), (\{a, b, c, d\}, 2), (\{h\}, 1), (\{f, h, g\}, 2)\}$ for $\mathcal{G}_1$, and $\mathcal{F}_2 = \{(\{a\}, 1), (\{g\}, 1), (\{c, d, e, f, g, h\}, 2)\}$ for $\mathcal{G}_2$. It then uses the greedy heuristic to solve min-HSP for input $(V,\, \mathcal{F}_1 \cup \mathcal{F}_2)$, which yields $M^{\text{join}} = \{a, g, h, b\}$ (b may be replaced by c or d). The refined placement based on initial placement V generates the same result. Since $\mathcal{G}_1$ already requires four monitors, both the joint and the refined placements are optimal in this example.

V. Optimality Conditions

We have seen from Section IV-C that, unlike the monitor placement problem in a static network, optimal monitor placement in a dynamic network is generally hard to compute. This motivates us to identify conditions under which the problem can be solved optimally. In this section, we investigate several such conditions in the order of increasing generality and identify the optimal solution in each case[8].

A. Optimality Condition for One-Shot Placement

A lower bound on the number of monitors needed by a robust placement is the maximum number of monitors placed by MMP in any one of all the possible topologies. Therefore, if the number of monitors needed to identify the base graph $\mathcal{G}_b$ matches the lower bound, the one-shot placement is guaranteed to be optimal.

Theorem V.1. If $\exists t \in \{1, \ldots, T\}$ such that the number of monitors placed by MMP in $\mathcal{G}_t$ equals the number of monitors placed by MMP in $\mathcal{G}_b$, then the one-shot placement is optimal.

Remark: A special case of this condition is when $\mathcal{G}_b = \mathcal{G}_t$ for some t, i.e., $\mathcal{G}_t$ is a subgraph of all other topologies $\mathcal{G}_{t'}$ for $t' \in \{1, \ldots, T\} \setminus t$.

B. Optimality Condition for Incremental Placement

Generally, the performance of incremental placement is sensitive to the order of applying IMMP to different topologies. In a special case where all topologies are 3-vertex-connected, however, the order no longer matters, and the incremental placement is guaranteed to be optimal.

Theorem V.2. If $\mathcal{G}_1, \ldots, \mathcal{G}_T$ are all 3-vertex-connected, then the incremental placement is optimal regardless of the order of processing the topologies.

Remark: Under this condition, the incremental (and optimal) placement is to simply place monitors at three arbitrary nodes.

C. Optimality Condition for Joint Placement

As explained in Section IV-C2, the difficulty in solving the monitor placement problem optimally is caused by the hardness of min-HSP. Therefore, any condition that allows min-HSP to be solved (optimally) in polynomial time is an

[7]Let $\mathcal{R}_v := \{(S_i, k_i) \in \mathcal{R} : v \in S_i\}$. Then it can be shown that the max-HSP on input $(M_0,\, \mathcal{R})$ is a maximization of a modular function subject to p-independence constraints with $p = \max_{v \in M_0} |\mathcal{R}_v|$. By [17], the number of redundant monitors found by the greedy heuristic is at least $1/p$ times the maximum number of redundant monitors.

[8]Note that the conditions are sufficient but not necessary, i.e., a solution may be optimal even if the corresponding optimality condition does not hold.

optimality condition for a polynomial-time joint placement algorithm. To this end, we establish a condition for efficiently solving min-HSP that generalizes a well-known condition for solving HSP and present a polynomial-time algorithm to provide an optimal solution when this condition is satisfied.

It is well known that HSP is equivalent to the *set cover problem (SCP)*. Naturally, min-HSP can also be represented by a variation of SCP, known as the *set multi-cover problem (SMCP)* [18]. Given an input $(U,\ \mathcal{E})$ for min-HSP, where $\mathcal{E} = \{(S_i,\ k_i) :\ S_i \subseteq U\}$ means that at least k_i items must be selected from S_i, we can construct an input $(\tilde{U},\ \tilde{\mathcal{E}})$ for SMCP, where $\tilde{U} = \mathcal{E}$ specifies each "item" S_i and its minimum frequency of coverage k_i, and $\tilde{\mathcal{E}} = \{\mathcal{S}_e :\ e \in U\}$ specifies the "sets" used to cover S_i's ($\mathcal{S}_e := \{S_i :\ e \in S_i\}$). The min-HSP is equivalent to SMCP that selects the minimum number of sets $\mathcal{E}' \subseteq \tilde{\mathcal{E}}$ to cover each S_i at least k_i times. We will work on SMCP for ease of presentation.

SMCP is NP-hard and can only be approximated within a factor of $(1 + \max_{e \in U} \log |\mathcal{S}_e|)$ by the greedy heuristic [16]. For inputs satisfying certain properties, however, SMCP can be solved optimally in polynomial time.

It is known that SCP is polynomial-time solvable when its input satisfies a condition known as the *consecutive ones property (C1P)* [19]. In words, a SCP has C1P if there is a permutation of the items such that each set covers a set of consecutive items. It can be shown that SMCP is also polynomial-time solvable under C1P[9]. This condition is, however, too strong. For example, the SMCP corresponding to monitor placement in a single topology can violate C1P (see an example in [15]), although the optimal placement is always polynomial-time solvable (by MMP).

Below, we give a more general condition for solving SMCP. Our condition is motivated by the following observation: for an item belonging to nested sets, i.e., each set is a subset of another, covering it by the set with the largest cardinality is always optimal as this set maximizes the coverage of other items. This obsrevation inspires the following condition.

Theorem V.3. If the items can be represented by nodes in a rooted tree such that each set covers a set of consecutive nodes along a leaf-to-root path in the tree, then SMCP (and the corresponding min-HSP) is polynomial-time solvable.

We prove this result by constructing an algorithm that solves SMCP optimally under the above condition. Let $\mathcal{I} = \{(e, k_e) :\ e \in U\}$ denote the items and their required frequency of coverage, and $\mathcal{E} = \{S_i :\ S_i \subseteq U\}$ the sets. The algorithm, referred to as *Leaf-based Greedy Cover (LGC)*, works as follows: while $\exists$ a non-sufficiently covered item (i.e., e such that the number of selected sets covering e is less than k_e),

1) for an arbitrary, non-sufficiently covered leaf item[10] e, select the set S that covers the most non-sufficiently covered items among sets covering e;
2) update the remaining frequency of coverage required by each item and the remaining sets.

[9]This is because under C1P, the constraint matrix for the ILP representation of SMCP is totally unimodular, and thus the LP relaxation gives an integral solution which is optimal [19].

[10]Precisely, e is such that e is not sufficiently covered, but all items below e in the rooted tree are sufficiently covered.

The idea of the proof is to show that under the condition in Theorem V.3, LGC is optimal; see [15] for details.

Translated back to the joint placement problem, the condition in Theorem V.3 requires that the sets of candidate monitors computed by FMP be representable by nodes in a rooted tree such that only consecutive sets on the same leaf-to-root path may overlap. One sufficient condition is as follows.

Corollary V.4. Let $\mathcal{F} = \{(S_i,\ k_i),\ i = 1, 2, \ldots\}$ be the constraints computed by FMP for $\{\mathcal{G}_1, \ldots, \mathcal{G}_T\}$. If for any $i \neq j$, S_i and S_j are either disjoint or nested (one is a subset of the other), then the optimal monitor placement for $\{\mathcal{G}_1, \ldots, \mathcal{G}_T\}$ can be computed in polynomial time.

Remark: Alternatively, the result of Corollary V.4 can be proved using the theory of minimizing a monotone concave cost function under laminar covering constraints [20]. Specifically, our cost function (number of monitors) is linear, and our constraints S_i's form a *laminar family* under the condition in Corollary V.4. Thus, we can apply the algorithm in [20] (for F_3) to select monitors optimally in $O(|V| \log^2 |V|)$ time.

A special case satisfying the condition in Corollary V.4 is that of placing monitors in a single topology, where S_i's are internal nodes in tri/bi/1-connected components of the topology. In this sense, Corollary V.4 provides an explanation on why computing the optimal monitor placement for a single topology is easy (solved by MMP) but computing that for multiple topologies is generally hard. In general, this condition holds if there is only splitting or merging of components during topology changes. For example, in Fig. 2, $\mathcal{G}_1$ contains a single triconnected component that is split into two in $\mathcal{G}_2$, and the monitor placement constraint is $\mathcal{F} = \{(\{a, b, c\}, 1), (\{f, g, h\}, 1), (\{a, b, c, d, e, f, g, h\}, 3)\}$, which satisfies the condition in Corollary V.4.

VI. Handling Practical Challenges

So far, we have focused on selecting monitors based on a given set of topologies. In practice, a robust monitoring system has to address additional challenges. For example, how do we determine the set of topologies to plan for? What if the planned monitor placement fails to identify every link at runtime? What if it requires too many monitors? What if nodes can appear/disappear? In this section, we briefly discuss each of these issues; more detailed discussions can be found in [15].

A. Topology Selection

Let $\{\mathcal{G}_t :\ t = 1, \ldots, N\}$ denote all possible topologies during the network lifetime and τ_t the expected fraction of time for the network to have topology $\mathcal{G}_t$. Assume that $\tau_1 \geq \tau_2 \geq \ldots$. Given a parameter τ denoting the required fraction of time that the network needs to be identifiable, the *topology selection problem* is to select a subset of topologies $\{\mathcal{G}_i :\ i \in I\}$ for $I \subseteq \{1, \ldots, N\}$ that can be identified with a minimum number of monitors, subject to the constraint that $\sum_{i \in I} \tau_i \geq \tau$.

The above optimization problem is clearly hard to solve as evaluating the objective function for a given solution (i.e., the minimum number of monitors to identify given topologies) is already NP-hard (see Theorem IV.5). A straightforward heuristic is to select the top-K frequently-occurring topologies for the minimum K such that $\sum_{t=1}^{K} \tau_t \geq \tau$. Alternatively, we can

select topologies in descending order of $\tilde{\tau}_t$, the sum frequency of all topologies containing $\mathcal{G}_t$ as subgraph, because a monitor placement that identifies $\mathcal{G}_t$ also identifies all topologies denser than $\mathcal{G}_t$ (see Section IV-A).

B. On-demand Monitor Placement

In case of unpredictable topology changes, existing monitors may not be able to identify all links. One way to ensure identifiability is to employ additional nodes as *temporary* monitors, which only participate in taking measurements but not in other functions such as storage or processing of measurements. To distinguish from temporary monitors, we refer to monitors placed during network planning as *persistent* monitors. Given the current topology $\mathcal{G}_t$ and the persistent monitors M_0 (placed by a robust monitor placement algorithm in Section IV), we can apply IMMP (Section IV-B) to select temporary monitors if needed to identify all links in $\mathcal{G}_t$.

C. Bounded Number of Monitors

Given a bounded number of persistent monitors, we generally have to employ temporary monitors to achieve identifiability at runtime. For system stability, it is desirable to minimize the number of times we change node status (monitor→non-monitor or non-monitor→monitor). When the topology changes from $\mathcal{G}_t$ to $\mathcal{G}_{t+1}$, the minimum number of nodes with changed status is upper-bounded by the minimum number of extra monitors (in addition to persistent monitors) to identify both $\mathcal{G}_t$ and $\mathcal{G}_{t+1}$, which is further upper-bounded by the minimum number of extra monitors to identify $\mathcal{G}_1, \ldots, \mathcal{G}_T$. The latter upper bound is minimized when the persistent monitors all belong to the optimal robust monitor placement. This observation suggests that we can adapt the algorithms in Section IV for placing a bounded number of persistent monitors by terminating the algorithms after selecting sufficient monitors.

D. Arrival/Departure of Nodes

We have limited topology changes to addition/removal of links, while in practice there may also be arrival/departure of nodes. We can handle arrivals by always selecting the newly arrived nodes as monitors; we can handle departures by treating each departure as removal of all the links incident to the departed node. It is easy to verify that this approach guarantees identifiability under node changes; we leave further optimization to future work.

VII. Performance Evaluation

We evaluate the proposed solutions on dynamic topologies extracted from mobility traces. We use two datasets: (1) taxi cab traces from San Francisco[11], from which we select traces of 86 nodes over a 8-hour period with location updates roughly every minute; (2) mobility traces generated by Rommie Hardy and Anjuli Smith at the Network Science Research Laboratory of the US Army Research Laboratory [21], which contain traces of 90 nodes belonging to 7 groups during a 400-second tactical operation with location updates every second[12].

[11]Traces are available at: http://crawdad.org/epfl/mobility/.

[12]The anonymized traces used for this evaluation are available at: https://www.dropbox.com/s/bkhdifos9wzjwln/trace_90node_401second.txt?dl=0

TABLE I
Dynamic Topologies for Taxi Network (86 nodes)

range (m)	#topology changes	avg #links	avg #components
500	479	95.1	31.3
1000	479	334.3	6.3
1500	479	694.8	2.5
2000	479	1106.5	1.5
2500	479	1528.3	1.1
3000	479	1934.0	1.0
3500	479	2286.8	1.0

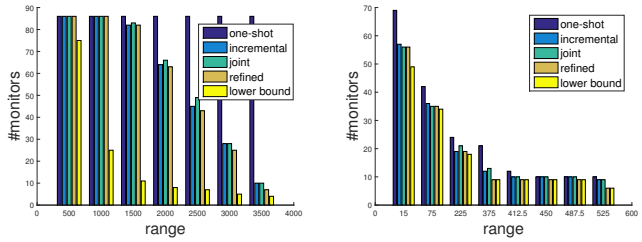
TABLE II
Dynamic Topologies for Tactical Network (90 nodes)

range (m)	#topology changes	avg #links	avg #components
15	399	325.9	17.3
75	293	539.9	10.4
225	196	1027.7	4.2
375	387	1256.1	2.0
450	380	1607.5	1.5
525	399	2191.1	1.1

Dataset (1) represents independent node mobility, and dataset (2) represents grouped node mobility.

We extract dynamic topologies from each trace by assuming a communication range and connecting two nodes by a link whenever they are within the range. See Tables I and II for a summary of extracted topologies. We see that both networks experience hundreds of topology changes during their lifetime. As expected, the network becomes denser (with a larger number of links) and better connected (with a smaller number of connected components) as the range increases.

Comparison of algorithms: We compare the performance of the proposed robust monitor placement algorithms in terms of the number of monitors, since all the algorithms guarantee identifiability. Here refined placement uses the result of one-shot placement as the initial set of monitors. We also evaluate a lower bound on the number of monitors by computing the maximum number of monitors placed by MMP in any single topology. Fig. 4 (a) shows the result for the taxi network and Fig. 4 (b) shows the result for the tactical network. As expected, the one-shot placement algorithm uses the most monitors and the refined placement algorithm uses the fewest. Comparing the two networks, we see that as the taxi network has very different topologies during its lifetime, it requires a large number of monitors to maintain identifiability, while the tactical network contains subnets with relatively stable topologies and thus requires fewer monitors. In particular, while the one-shot placement performs much worse than the others for the taxi network, it performs near-optimally for the tactical network as the communication range increases.

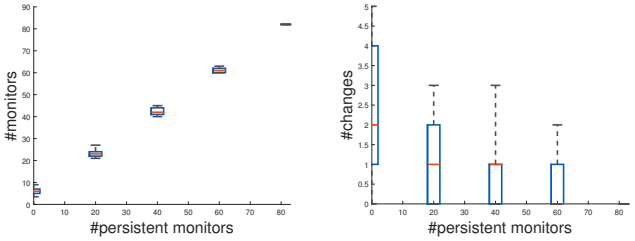


(a) taxi (b) tactical

Fig. 4. Performance comparison under varying communication range.

Varying number of persistent monitors: Fig. 5 shows

the results for the taxi network when we bound the number of persistent monitors and place temporary monitors as needed to achieve identifiability, where the persistent monitors are randomly selected from monitors placed by the refined placement algorithm. To evaluate the cost of adapting monitor placement, we count the number of changes in node status, where a change occurs when a non-monitor is selected as a temporary monitor or a temporary monitor becomes a non-monitor. Fig. 5 (a) shows the median (red bar), 25/75-th percentile (box), and 5/95-th percentile (whisker) of the number of (both persistent and temporary) monitors over all topologies, and Fig. 5 (b) shows the corresponding values for the number of node status changes. The results show a clear tradeoff between the cost of monitors and the cost of adaptation, controlled by the number of persistent monitors. We have similar observations for the tactical network; see [15].



(a) #monitors (b) #changes

Fig. 5. Varying persistent monitors (taxi network, range= 1500 m).

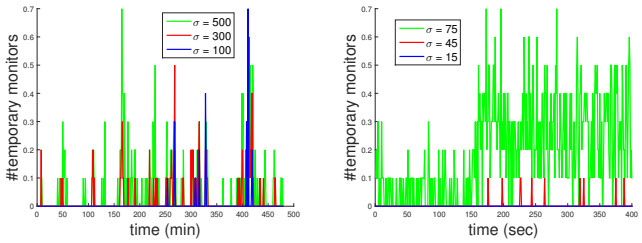
Impact of prediction error: To evaluate how well our placement algorithms perform when input topologies contain errors, we add i.i.d. zero-mean Gaussian noise with variance σ^2 to node locations (x/y-coordinates) and treat the resulting topologies as the new ground truth; the process is repeated for multiple Monte Carlo runs. Meanwhile, the topologies computed from the original trace are provided to a monitor placement algorithm (refined placement) as predicted topologies[13]. We evaluate the robustness of the resulting placement by: (i) testing whether the placement achieves identifiability for each newly generated topology, and (ii) if not, computing the number of temporary monitors needed to achieve identifiability. As shown in Fig. 6, our monitor placement is highly robust to prediction error, requiring little help from temporary monitors even if the error in node location prediction[14] is as large as $1/3$ of the communication range. In fact, our placement achieves identifiability (i.e., not requiring any temporary monitor) for 0.95 fraction of time for the taxi network and 0.82 fraction of time for the tactical network (not shown).

VIII. Conclusion

We have studied the problem of placing monitors to identify additive link metrics from end-to-end measurements in the face of topology changes. Unlike existing monitor placement algorithms that consider a single topology, our algorithms maintain identifiability across multiple topologies. By casting the problem as a generalized hitting set problem, we show that the optimal placement is NP-hard, but can be approximated

[13]This is equivalent to treating the original topologies as ground truth and the modified topologies as estimates.

[14]The value of σ is basically the root-mean-squared error (RMSE) of the maximum likelihood estimate of node locations.



(a) taxi (range = 1500 m) (b) tactical (range = 225 m)

Fig. 6. Number of temporary monitors (averaged over 10 Monte Carlo runs) needed to achieve identifiability under prediction error.

to a logarithmic factor by the greedy heuristic and can be solved exactly in several special cases. We further extend the solution to handle several practical challenges. Our evaluations on dynamic topologies driven by mobility traces verify the effectiveness and robustness of the proposed solution.

References

[1] Y. Vardi, "Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Assoc.*, pp. 365–377, 1996.

[2] A. B. Downey, "Using pathchar to estimate internet link characteristics," in *IEEE SIGCOMM*, 1999.

[3] G. Jin, G. Yang, B. Crowley, and D. Agarwal, "Network characterization service (NCS)," in *IEEE HPDC*, 2001.

[4] E. Lawrence and G. Michailidis, "Network tomography: A review and recent developments," *Frontiers in Statistics*, vol. 54, 2006.

[5] O. Gurewitz and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," in *IEEE INFOCOM*, 2001.

[6] Y. Chen, D. Bindel, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *ACM SIGCOMM*, 2004.

[7] A. Chen, J. Cao, and T. Bu, "Network Tomography: Identifiability and Fourier domain estimation," in *IEEE INFOCOM*, 2007.

[8] A. Gopalan and S. Ramasubramanian, "On identifying additive link metrics using linearly independent cycles and paths," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, 2012.

[9] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, "Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1351–1368, June 2014.

[10] M. Zhao and W. Wang, "Analyzing topology dynamics in ad hoc networks using a smooth mobility model," in *IEEE WCNC*, 2007.

[11] I. W. Ho, K. K. Leung, and J. W. Polak, "Stochastic model and connectivity dynamics for vanets in signalized road systems," *IEEE/ACM Transactions on Networking*, vol. 19, pp. 195–208, 2011.

[12] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *IEEE INFOCOM*, 2004.

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

[14] R. Kumar and J. Kaur, "Practical beacon placement for link monitoring using network tomography," *IEEE JSAC*, vol. 24, 2006.

[15] T. He, L. Ma, A. Gkelias, K. K. Leung, A. Swami, and D. Towsley, "Robust monitor placement for network tomography in dynamic networks," Technical Report, July 2015. [Online]. Available: http://researcher.watson.ibm.com/researcher/files/us-the/RobustMonitorPlacement.pdf

[16] G. Dobson, "Worst-case analysis of greedy heuristics for integer programming with non-negative data," *Mathematics of Operations Research*, vol. 7, no. 4, pp. 515–531, November 1982.

[17] B. Korte and D. Hausmann, "An analysis of the greedy heuristic for independence systems," *Annals of Discrete Mathematics*, vol. 2, pp. 65–74, 1978.

[18] S. Rajagopalan and V. Vazirani, "Primal-dual RNC approximation algorithms for (multi)-set (multi)-cover and covering integer programs," in *IEEE FOCS*, 1993.

[19] N. Rug and A. Schobel, "Set covering with almost consecutive ones property," *Discrete Optimization*, vol. 1, pp. 215–228, November 2004.

[20] M. Sakashita, K. Makino, and S. Fujishige, "Minimizing a monotone concave function with laminar covering constraints," *Discrete Applied Mathematics*, vol. 156, no. 11, pp. 2004–2019, June 2008.

[21] "The Network Science Research Laboratory," US Army Research Laboratory. [Online]. Available: https://www.arl.army.mil/www/default.cfm?page=2485