

IDENTIFIABILITY OF LINK METRICS IN COMMUNICATION NETWORKS: THEORY AND ALGORITHM DESIGNS

LIANG MA

A Thesis Submitted in Fulfilment of Requirements for the Degree of
Doctor of Philosophy of Imperial College London and
Diploma of Imperial College

Communications and Signal Processing Group
Department of Electrical and Electronic Engineering
Imperial College London

2014

The copyright of this thesis rests with the author and is made available under a
Creative Commons Attribution Non-Commercial No Derivatives licence.

Researchers are free to copy, distribute or transmit the thesis on the condition that
they attribute it, that they do not use it for commercial purposes and that they do not
alter, transform or build upon it. For any reuse or redistribution, researchers must
make clear to others the licence terms of this work.

Declaration of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Liang Ma

Abstract

In this thesis, we investigate the problem of identifying individual link metrics in a communication network from accumulated end-to-end metrics over selected measurement paths, under the assumption that link metrics are additive and constant during the measurement, and measurement paths cannot contain cycles. We know from linear algebra that all link metrics can be uniquely identified when the number of linearly independent measurement paths equals n , the number of links. It is, however, inefficient to collect measurements from all possible paths, whose number can grow exponentially in n , as the number of useful measurements (from linearly independent paths) is at most n . The aim of this thesis is thus to characterize network identifiability by easily verifiable conditions and develop efficient algorithms for achieving and maximizing network identifiability.

To characterize network identifiability in terms of the network topology and the number/placement of monitors, our main results are: (i) it is generally impossible to identify all the link metrics by using two monitors; (ii) nevertheless, metrics of all the interior links not incident to any monitor are identifiable by two monitors if the topology satisfies a set of necessary and sufficient connectivity conditions; (iii) these conditions naturally extend to a necessary and sufficient condition for identifying all the link metrics using three or more monitors. We show that these conditions not only allow efficient identifiability tests, but also enable efficient algorithm design for constructing linearly independent paths and computing individual link metrics. Specifically, we show that whenever there exists a set of n linearly independent measurement paths, there must exist a set of three pairwise independent spanning trees. We exploit this property to develop an algorithm that can construct n linearly inde-

pendent, cycle-free paths between monitors without examining all candidate paths, whose complexity is quadratic in n . A further benefit of the proposed algorithm is that the generated paths satisfy a nested structure that allows linear-time computation of link metrics without explicitly inverting the measurement matrix. Next, we study a complementary problem of how to characterize network partial identifiability when n linearly independent paths cannot be found in a given network, for which we establish an efficient algorithm to determine all identifiable links in an arbitrary network under a given monitor placement. Finally, we investigate a realistic problem of how to place monitors such that the network uncertainty with respect to internal link metrics is minimized. To this end, we first develop efficient algorithm to place the minimum number of monitors in order to identify all link metrics. Our evaluations on both random and real topologies show that the proposed minimum monitor placement algorithm achieves identifiability using a much smaller number of monitors than a baseline solution. However, we observe that the complete identification of all link metrics in a network can require a large number of monitors (e.g., 60% of all nodes), even if assuming optimal placement of monitors. Therefore, we then study the problem of placing a given number of monitors (κ monitors) to identify the maximum number of link metrics. For this problem, we build a polynomial-time algorithm to incrementally place monitors such that each newly placed monitor maximizes the number of additional identifiable links. The significance of this κ -monitor placement algorithm is that it is provable optimal if the network is 2-vertex-connected. For other types of networks, our evaluation on various ISP topologies shows that the proposed κ -monitor placement algorithm allows identification for close to the maximum number of links while incurring a much lower complexity than brute-force approaches.

Acknowledgments

First and foremost, I want to express my deepest appreciation to my Ph.D. supervisor Prof. Kin K. Leung for his enlightening guidance, treasurable suggestions, generous support, and fatherlike encouragement. He not only teaches me academically with his in-depth professional knowledge, but also shows me effective methodologies for solving complicated problems systematically. More importantly, Prof. Leung sets a true example to me on how a world-class researcher should behave, which I firmly believe is my lifetime standard to contribute the scientific research community.

I would like to thank Dr. Ting He, research scientist at IBM T. J. Watson Research Center New York, for her countless worthwhile comments and suggestions on my research work, and also hosting me two memorable summer internships at IBM Research New York in 2012–2013. Her impressively strong ability in mathematics and rigorous logical thinking becomes the valuable and powerful fuel that drives me to refine my research skills continuously. I would also like to thank Dr. Kang-Won Lee, who, as my manager during my summer internship at IBM, provided me with numerous golden opportunities to work with distinguished scientists in this world. Moreover, special thanks to Dr. Ananthram Swami at US Army Research Laboratory (ARL), Adelphi, USA, and Prof. Don Towsley at University of Massachusetts, Amherst, USA, for their immense effort in discussing detailed technical issues, commenting research results, and identifying potential future directions in my work. It is my great honor and true pleasure to collaborate with these prestigious and exemplary researchers from both industry and academia, without whom this thesis would not have been possible. This UK-USA collaboration is so appreciated, as it not only generated fruitful publications including one Best Paper Award and one Best Paper

Award Finalist in 2013, but also left unforgettable and precious memories to me.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. Throughout my Ph.D., I was funded by UK-USA ITA (International Technology Alliance) project, EPSRC (Engineering and Physical Sciences Research Council, UK), and EEE (Electrical and Electronic Engineering) departmental scholarships at Imperial College.

Thanks to Dr. Moez Draief from Imperial College London and Prof. Patrick Thiran from Swiss Federal Institute of Technology at Lausanne (École Polytechnique Fédérale de Lausanne, EPFL), Switzerland, in my viva examination committee for their valuable time and insightful questions.

I am also indebted to the wonderful colleagues and friends in the CSP (Communications and Signal Processing) group at Imperial College for making my years in London much more enjoyable than it would have been otherwise.

Lastly, I gratefully appreciate the most solid and fundamental energy source of my life: my parents. Thanks for their unconditional love, care, support, and encouragement. It is my parents who always encourage me to face up to life difficulties bravely and continue the rough road to fight for what I believe in. Without Mom and Dad, I would be nowhere near the achievements I got today. I would never be thankful enough to my parents for what they have done for me, but to say simple words - thank you!

Liang Ma

Imperial College London

May, 2014

Contents

Abstract	iii
Acknowledgments	vi
External Publications	xiii
List of Tables	xvi
List of Figures	xx
Abbreviations	xxi
Mathematical Notations	xxii
1 Introduction	1
1.1 Overview	1
1.2 Research Challenges and Objectives	3
1.2.1 Challenges	3
1.2.2 Objectives	4
1.3 Summary of Contributions	4
1.4 Thesis Organization	6
2 Literature Review and Problem Formulation in Network Tomography	7
2.1 Literature Review	7
2.1.1 Complete Network Identifiability	7
2.1.2 Path Construction and Link Identification	9
2.1.3 Partial Network Identifiability	9
2.1.4 Monitor Placement	10

2.2	Problem Formulation	11
2.2.1	Research Issues	13
2.2.2	Illustrative Example	15
3	Network Identifiability Conditions	17
3.1	Unidentifiability with Two Monitors	17
3.2	Identifiability of Interior Links with Two Monitors	21
3.2.1	Proof of Theorem 3.3: Necessary Part	23
3.2.2	Proof of Theorem 3.3: Sufficient Part	27
3.3	Identifiability with Three or More Monitors	32
3.3.1	Conversion into Two-Monitor Problem	32
3.3.2	Complete Identifiability Condition	34
3.4	Testing and Ensuring Identifiability	37
3.4.1	Efficient Test for Interior Graph Identifiability	37
3.4.2	Efficient Test for Complete Identifiability	38
4	Path Construction and Link Identification for Completely Identifiable Networks	40
4.1	Algorithm Overview	40
4.2	Spanning Tree-based Path Construction	42
4.3	Spanning Tree-based Link Identification	46
4.4	Complexity Analysis	48
4.5	An Example	50
4.6	Performance Evaluation	52
4.6.1	Random Topologies	54
4.6.2	ISP Topologies	58
5	Link Identification in Partially Identifiable Networks	60
5.1	Algorithm Overview	61

5.2	Algorithm Details	63
5.2.1	Decomposition into Triconnected Components	63
5.2.2	Classification of Triconnected Components	64
5.2.3	Determination of All Identifiable Links	70
5.2.4	Complexity Analysis	73
5.2.5	Example	73
6	Algorithms for Optimal Monitor Placement	75
6.1	Optimal Monitor Placement for Complete Network Identifiability	76
6.1.1	Minimum Monitor Placement (MMP)	77
6.1.2	Optimality of MMP	78
6.1.3	Complexity of MMP	79
6.1.4	Example	79
6.2	Maximal Identifiability Monitor Placement	80
6.2.1	Candidate Monitor Selection	80
6.2.2	Greedy Maximal identifiability Monitor Placement (GMMP)	83
6.2.3	Optimality of GMMP for 2-vertex-connected Networks	84
6.2.4	Complexity of GMMP	89
6.2.5	Example	90
6.3	Algorithm Evaluation	90
6.3.1	Evaluation of MMP	90
6.3.2	Evaluation of GMMP	99
7	Conclusions and Future Work	104
7.1	Conclusions	104
7.2	Future Work	106
	Bibliography	108

A	Auxiliary Algorithms	115
B	Independent Spanning Trees	119
C	Graph Decompositions	122
D	Supporting Theorems	123
D.1	Theorems	123
D.2	Proofs	125
D.2.1	Proof of Lemma D.1	125
D.2.2	Proof of Proposition D.2	126
D.2.3	Proof of Lemma D.3	126
D.2.4	Proof of Claim 1	130
D.2.5	Proof of Lemma D.4	130
D.2.6	Proof of Lemma D.5	132
D.2.7	Proof of Lemma D.6	135
D.2.8	Proof of Lemma D.7	137
D.2.9	Proof of Theorem D.8	139
E	Theorem Proof	142
E.1	Proof of Lemma 3.4	142
E.2	Proof of Lemma 3.5	148
E.2.1	Proof of Lemma 3.5-(a)	148
E.2.2	Proof of Lemma 3.5-(b)	152
E.3	Proof of Theorem 4.2	154
E.3.1	Number of Linearly Independent Paths After Removing All Virtual Links	155
E.3.2	Number of Distinct Paths by STPC	160
E.4	Proof of Theorem 5.3	163

E.5 Proof of Theorem 6.1	172
------------------------------------	-----

External Publications

List of publications arising directly from this thesis

- **Journal Papers**

1. **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,” *accepted to IEEE/ACM Transactions on Networking*, May, 2014.

- **Conference Papers**

1. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Monitor Placement for Maximal Identifiability in Network Tomography,” in *Proc. of IEEE INFOCOM*, 2014, Toronto, Canada.
2. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Link Identifiability in Communication Networks with Two Monitors,” in *Proc. of IEEE Globecom*, 2013, Atlanta, U.S.
3. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Identifiability of Link Metrics Based on End-to-end Path Measurements,” in *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, 2013, Barcelona, Spain. **(Best Paper Award Finalist)**
4. **L. Ma**, T. He, K. K. Leung, D. Towsley, and A. Swami, “Efficient Identification of Additive Link Metrics via Network Tomography,” in *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, 2013, Philadelphia, U.S. **(Best Paper Award)**
5. **L. Ma**, T. He, K. K. Leung, D. Towsley, and A. Swami, “Efficient Identification of Additive Link Metrics via Network Tomography,” in *Proc. of Annual*

Fall Meeting of International Technology Alliance, 2013, New York, U.S.

6. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Link Identifiability in Communication Networks with Two Monitors,” in *Proc. of Annual Fall Meeting of International Technology Alliance*, 2013, New York, U.S.
7. **L. Ma**, T. He, K. K. Leung, D. Towsley, and A. Swami, “Topological Constraints on Identifying Additive Link Metrics via End-to-end Paths Measurements,” in *Proc. of The Annual Conference of The International Technology Alliance (ACITA)*, 2012, Southampton, U.K.

- **Patents**

1. **L. Ma**, C. Bisdikian, T. He, and B. Ray, “Bounded-Budget Monitor Deployment in Monitoring Networks via End-to-end Probes,” *IBM*, July, 2013.

- **Technical Reports**

1. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Link Identifiability with Two Monitors: Auxiliary Algorithms and Proof of Selected Theorems,” *Technical Report*, Imperial College, London, UK, Aug., 2013.
2. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Partial Network Identifiability: Theorem Proof and Evaluation,” *Technical Report*, Imperial College, London, UK, Jul., 2013.
3. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Link Identifiability with Two Monitors: Proof of Selected Theorems,” *Technical Report*, Imperial College, London, UK, Feb., 2013.
4. **L. Ma**, T. He, K. K. Leung, D. Towsley, and A. Swami, “Efficient Identification of Additive Link Metrics: Theorem Proof and Evaluations,” *Technical Report*, Imperial College, London, UK, Nov., 2012.
5. **L. Ma**, T. He, K. K. Leung, A. Swami, and D. Towsley, “Identification of Additive Link Metrics: Proof of Selected Theorems,” *Technical Report*, Imperial

College, London, UK, July, 2012.

6. **L. Ma**, T. He, K. K. Leung, D. Towsley, and A. Swami, “Additive Link Metrics Identification: Proof of Selected Lemmas and Propositions,” *Technical Report*, Imperial College, London, UK, May, 2012.

Other publications

• Conference Papers

1. **L. Ma**, T. He, A. Swami, K.-W. Lee, and K. K. Leung, “Switch-and-Navigate: Controlling Data Ferry Mobility for Delay-Bounded Messages,” in *Proc. of IEEE MILCOM*, 2011, Nov. 7-10, Baltimore, Maryland, U.S.
2. **L. Ma**, T. He, A. Swami, K.-W. Lee, and K. K. Leung, “Hierarchical Mobility Control for Data Ferries under Constrained Message Delays,” in *Proc. of The Annual Conference of The International Technology Alliance (ACITA)*, 2011, Maryland, U.S.

List of Tables

4.1	Paths for Identifying Tree Links	50
4.2	Paths for Identifying Non-tree Links	52
4.3	Random Graphs (ER: $p = 0.0656$, RG: $d_c = 0.15554$, BA: $n_{\min} = 5$, $I_{\max} = 3 \times n$)	56
4.4	ISP Topologies ($I_{\max} = 8 \times n$ for the first 5 networks, and $I_{\max} = \infty$ for the last 3)	56
6.1	Parameters of AS Topologies in Rocketfuel	96
6.2	Parameters of AS Topologies in CAIDA	99
6.3	ISP Networks for Evaluation of GMMP	100

List of Figures

2.1	Sample network with given monitor placement: (a) two monitors m_1 and m_2 ; (b) three monitors m_1, m_2 , and m_3	15
3.1	Organizing graph $\mathcal{G}$ into exterior links and interior graph ($\{a_i\}_{i=1}^{k_1}$ and $\{b_j\}_{j=1}^{k_2}$ may have overlap).	18
3.2	Illustration of Condition ①, where $\{l_1, l_2\}$ is an edge cut in (c) and (d).	24
3.3	Possible scenarios of a cut-vertex r	25
3.4	Possible scenarios of a 2-vertex cut $\{r, s\}$	26
3.5	Two types of identifiable links in $\mathcal{H}$	27
3.6	Sample network with identifiable interior graph.	29
3.7	Possible cases of interior link vw : (a) Case A, (b) Case B-1, (c) Case B-2 (Case B-2 is independent of the two monitor locations).	29
3.8	(a) $\mathcal{G}$ with κ ($\kappa \geq 3$) monitors; (b) $\mathcal{G}_{ex}$ with two virtual monitors: m_1^+ and m_2^+	33
3.9	$\mathcal{G}_{ex} - v_1 - v_2$ is disconnected, where $v_1, v_2 \in V(\mathcal{G})$	36
4.1	The r -extended graph $\mathcal{G}_{ex}^*$ of $\mathcal{G}$	40
4.2	Three internally vertex disjoint paths between v and r imply three monitor-to-monitor simple paths: $\mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{S}_1 \cup \mathcal{S}_3$, and $\mathcal{S}_2 \cup \mathcal{S}_3$	41
4.3	Constructing measurement path traversing a non-tree link vw	44
4.4	Sample network and its r -extended graph.	49
4.5	Spanning trees of $\mathcal{G}_{ex}^*$ in Fig. 4.4 (b), independent w.r.t. r (node 1). Here arrows are only used to illustrate paths to r ; the original links are undirected.	49
5.1	Three cases in identifying links of a 3-vertex-connected sub-graph $\mathcal{D}$	60

5.2	Decomposition of a sample network: (a) sample network $\mathcal{G}$; (b) bi-connected components of $\mathcal{G}$; (c) triconnected components of $\mathcal{G}$	62
5.3	Agent: v_1 and v_2 are agents w.r.t. $\mathcal{B}$, but v_3 is not an agent w.r.t. $\mathcal{B}$ (m_1, m_2 : monitors).	65
5.4	Biconnected component $\mathcal{B}$ with 3 or more agents.	66
5.5	Agents m'_1 and m'_2 wrt biconnected component $\mathcal{B}$	66
5.6	Type- k vertex cut: $\mathcal{T}_1, \dots, \mathcal{T}_6$ are triconnected components within the same biconnected component, for which m'_1, m'_2, m'_3 are agents.	68
6.1	(a) Original graph; (b) biconnected components; (c) triconnected components.	76
6.2	Example: $\mathcal{O}_2 = \{v_1, v_2\}$ is not a sub-set of $\mathcal{O}_3 = \{v_a, v_b, v_c\}$	84
6.3	Optimal monitor placement in 2-vertex-connected networks.	85
6.4	Nested structure of optimal monitor placement in 2-vertex-connected networks.	87
6.5	Monitor placement by GMMP in a sample network: (a) $\kappa = 3$; (b) $\kappa = 4$; (c) $\kappa = 5$, where $\mathcal{O}_3 \subset \mathcal{O}_4 \subset \mathcal{O}_5$	90
6.6	Comparison between RMP and MMP: densely-connected graphs ($\bar{n} = 441, \bar{\kappa}_{\text{MMP}} = 3$ for BA, $\bar{n} = 437, \bar{\kappa}_{\text{MMP}} = 9.36$ for ER, $\bar{n} = 437, \bar{\kappa}_{\text{MMP}} = 19.42$ for PL and $\bar{n} = 451, \bar{\kappa}_{\text{MMP}} = 14.52$ for RG).	92
6.7	Comparison between RMP and MMP: sparsely-connected graphs ($\bar{n} = 295, \bar{\kappa}_{\text{MMP}} = 73.51$ for BA, $\bar{n} = 293, \bar{\kappa}_{\text{MMP}} = 36.76$ for ER and $\bar{n} = 297, \bar{\kappa}_{\text{MMP}} = 40.24$ for PL).	92
6.8	Comparison between RMP and MMP: ISP topologies in Rocketfuel ($15 \cdot V $ Monte Carlo runs, $r_{\text{MMP}} := \kappa_{\text{MMP}}/ V $).	97
6.9	Comparison between RMP and MMP: AS topologies in CAIDA (3000 Monte Carlo runs, $r_{\text{MMP}} := \kappa_{\text{MMP}}/ V $).	97

6.10	Identification ratio in ISP networks; $\star : (\kappa_{\text{MMP}}/ V , 1)$, i.e., transition point for complete identification.	101
6.11	Algorithm running time in ISP networks.	102
A.1	Four sub-categories of Category 2.	116
D.1	Two cases of bridge link l : (a) interior bridge, (b) exterior bridge. . .	123
D.2	Paths generated by merging red and green path segments.	127
D.3	Paths generated by merging blue and red/green path segments. . . .	128
D.4	Virtual link replacement.	130
D.5	Triangle identification (v_1 is an agent).	132
D.6	Triangle identification (v_1v_2 is a cross-link).	133
D.7	Triangle identification (there exists $\mathcal{T}_1^*$ in $\mathcal{B}_1$).	134
D.8	Link identifications of Category 2.3.	136
D.9	Category 2.3 - unidentifiable $\mu_2\mu_3$	136
D.10	Direct link $\mu_1\mu_2$ identification.	138
D.11	Direct link $\mu_1\mu_2$ identification in the abstracted graph.	138
D.12	Decomposition of triconnected components in a polygon.	140
E.1	Two cycles with four common nodes.	143
E.2	Replacement operation to obtain an induced cycle.	144
E.3	Counter example - a non-separating cycle containing vw cannot be found.	144
E.4	Construction of two cycles and two paths.	145
E.5	Case-B links vw and xy cannot be in the same non-separating cycle. . . .	149
E.6	Case B-2 link vw and monitors are not in the same non-separating cycle.	152
E.7	Paths construction for Case-B link vw	153
E.8	Combining red and green paths w.r.t. m_1	154

E.9 Path construction w.r.t. m_2	156
E.10 Path construction w.r.t. m_a	156
E.11 Path construction w.r.t. m_b	157
E.12 Duplicate paths w.r.t. m_1 and m_2^+	157
E.13 Duplicate paths w.r.t. m_1^+	158
E.14 Other virtual links connecting to m_1^+ or m_2^+	158
E.15 Shortening a long path in Y'	161
E.16 Link identifications of Category 1.	164
E.17 Link identifications of Category 2.1.	165
E.18 Link identifications of Category 2.2.	166
E.19 Link identifications of Category 2.4.	168
E.20 Link identification of Category 3.	168
E.21 Link identification of Category 2.2 in different triconnected components.	170
E.22 Link identification of Category 3 in different triconnected components.	170
E.23 Possible scenarios for a triconnected component $\mathcal{T}$ with $s_{\mathcal{T}} = 1$	174
E.24 Possible scenarios for a triconnected component $\mathcal{T}$ with $s_{\mathcal{T}} = 2$	174
E.25 Possible scenarios for a triconnected component $\mathcal{T}$ with $s_{\mathcal{T}} \geq 3$	175

Abbreviations

AS	Autonomous System
ARMP-$\mathcal{S}$	Advanced Random Monitor Placement - from candidate set $\mathcal{S}$
ARMP-V	Advanced Random Monitor Placement - from node set V
DAIL	Determination of All Identifiable Links
GMMP	Greedy Maximal identifiability Monitor Placement
ICMP	Internet Control Message Protocol
ISP	Internet Service Provider
MILI	Matrix Inversion-based Link Identification
MMP	Minimum Monitor Placement
NCS	Network Characterization Service
RMP	Random Monitor Placement
RWPC	Random Walk-based Path Construction
SDN	Software-Defined Networking
STLI	Spanning Tree-based Link Identification
STPC	Spanning Tree-based Path Construction
TTL	Time-To-Live

Mathematical Notations

$V(\mathcal{G}), L(\mathcal{G})$	set of nodes/links in graph $\mathcal{G}$
$ \mathcal{G} $	degree of graph $\mathcal{G}$: $ \mathcal{G} = V(\mathcal{G}) $ (number of nodes)
$ \mathcal{G} $	order of graph $\mathcal{G}$: $ \mathcal{G} = L(\mathcal{G}) $ (number of links)
n	number of links in $\mathcal{G}$
$\mathcal{H}$	interior graph (see Definition 1)
$L(v)$	set of links incident to node v
$\mathcal{G} - l$	delete a link: $\mathcal{G} - l = (V(\mathcal{G}), L(\mathcal{G}) \setminus \{l\})$, where $l \in L(\mathcal{G})$ and “ $\setminus$ ” is setminus
$\mathcal{G} + l$	add a link: $\mathcal{G} + l = (V(\mathcal{G}), L(\mathcal{G}) \cup \{l\})$, where the end-points of link l are in $V(\mathcal{G})$
$\mathcal{G} - v$	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 sub-graph of $\mathcal{G}$, $v \in V(\mathcal{G}) \setminus V(\mathcal{G}_s)$, and L_v is the set of all links between v and nodes in $V(\mathcal{G}_s)$
$\mathcal{G} \setminus \mathcal{G}'$	from $\mathcal{G}$, delete all common nodes with $\mathcal{G}'$ and their incident links
$\mathcal{G} \cap \mathcal{G}'$	intersection of graphs: $\mathcal{G} \cap \mathcal{G}' = (V(\mathcal{G}) \cap V(\mathcal{G}'), L(\mathcal{G}) \cap L(\mathcal{G}'))$
$\mathcal{G} \cup \mathcal{G}'$	union of graphs: $\mathcal{G} \cup \mathcal{G}' = (V(\mathcal{G}) \cup V(\mathcal{G}'), L(\mathcal{G}) \cup L(\mathcal{G}'))$
$\mathcal{P}(v_0, v_k)$	simple path connecting nodes v_0 and v_k , defined as a special graph with $V(\mathcal{P}) = \{v_0, \dots, v_k\}$ and $L(\mathcal{P}) = \{v_0v_1, v_1v_2, \dots, v_{k-1}v_k\}$
$\mathcal{C}$	cycle: if $(v_0, \dots, v_k)$ ($k \geq 2$) is a sequence of nodes on a simple path $\mathcal{P}$, then $\mathcal{C} = \mathcal{P} + v_kv_0$ is a <i>cycle</i>
$\mathcal{F}$	a non-separating cycle (see Definition 6)

$W_l, W_{\mathcal{P}}$	metric on link l and sum metric on path $\mathcal{P}$
m_i	$m_i \in V(\mathcal{G})$ is the i -th monitor in $\mathcal{G}$
m_i^+	the i -th virtual monitor in an extended graph $\mathcal{G}_{ex}$
m'_i	the i -th agent in a biconnected component of $\mathcal{G}$ (see Definition 9)
μ_i	the i -th vantage in a triconnected component of $\mathcal{G}$ (see Definition 11)
$N_{\mathcal{B}}, N_{\mathcal{T}}$	number of biconnected/triconnected components in $\mathcal{G}$
κ_{MMP}	minimum number of monitors for complete network identification computed by MMP (see Section 6.1.1)
r_{MMP}	$\kappa_{\text{MMP}}/ V(\mathcal{G}) $
$\mathcal{S}$	candidate set for monitor placement in graph $\mathcal{G}$
$\mathcal{O}_{\kappa}$	κ -monitor placement
$\mathcal{O}_{\kappa}^*$	optimal κ -monitor placement
$N(\mathcal{O})$	number of identifiable links for monitor placement $\mathcal{O}$ ($\mathcal{O} \subseteq V(\mathcal{G})$)
$\mathcal{G}_m$	the union of three independent spanning trees in an r -extended graph $\mathcal{G}_{ex}^*$
$v_0 \underline{a_1 \cdots a_k} v_k$	path $\mathcal{P}(v_0, v_k)$, where $\underline{a_1 \cdots a_k}$ is part of the sequenced intermediate nodes or labeled letters along the v_0 to v_k direction, is denoted by $v_0 \underline{a_1 \cdots a_k} v_k$
$\mathfrak{S}_{\mathcal{P}}$	the sequence of nodes $v_0 v_1 \dots v_k$ along path $\mathcal{P}$
$\overset{\circ}{\mathcal{P}}$	given path $\mathcal{P}$ with $\mathfrak{S}_{\mathcal{P}} = v_0 v_1 \dots v_k$, $\overset{\circ}{\mathcal{P}}$ is a sub-path with $\mathfrak{S}_{\overset{\circ}{\mathcal{P}}} = v_1 \dots v_{k-1}$
$\mathcal{P}_{\overset{\circ}{v_i}}$	given path $\mathcal{P}$ with $\mathfrak{S}_{\mathcal{P}} = v_0 v_1 \dots v_k$, $\mathcal{P}_{\overset{\circ}{v_i}}$ is a sub-path with $\mathfrak{S}_{\mathcal{P}_{\overset{\circ}{v_i}}} = v_0 \dots v_{i-1}$ ($i \in \{0, \dots, k\}$)
$\overset{\circ}{v_i} \mathcal{P}$	given path $\mathcal{P}$ with $\mathfrak{S}_{\mathcal{P}} = v_0 v_1 \dots v_k$, $\overset{\circ}{v_i} \mathcal{P}$ is a sub-path with $\mathfrak{S}_{\overset{\circ}{v_i} \mathcal{P}} = v_{i+1} \dots v_k$ ($i \in \{0, \dots, k\}$)

Introduction

1.1 Overview

Modern communication networks, hybrid in nature, inherently contain multiple sub-networks due to the presence of (commercial) partners, mobile devices, and third-party infrastructures (e.g., cellular networks). Though exhibiting high heterogeneity and complexity, hybrid networks can provide a communication substrate needed to support services that are vital to daily network operations. This substrate is therefore required to be reliable and robust to disruptions. In order to provide these qualities, it is essential to have accurate and timely knowledge of network internal states, e.g., delays on individual links. Moreover, when abnormal behavior is detected, it is crucial to diagnose and locate the source of the abnormality. In the case of service disruption, knowledge of where the abnormal source resides can enable fast service recovery by migrating the affected services and/or routing around the affected region. However, direct performance measurement of network internal states (e.g., link delay) is not always feasible due to the traffic overhead of the measurement process or the lack of protocol support at network internal elements for making such measurements [1]. Therefore, this poses challenges on how to reveal network internal states with minimum internal support.

One effective scheme to address this issue is by employing *external* approaches, where we infer network internal states by measuring the performance along selected paths from a sub-set of nodes with monitoring capabilities, hereafter referred to as

monitors.

Depending on the granularity of observations, external approaches can be classified as *hop-by-hop* approaches or *end-to-end* approaches. The former rely on special diagnostic tools such as *traceroute*, *pathchar* [2], and *Network Characterization Service (NCS)* [3] to reveal fine-grained performance metrics of individual links by sending active probes. Traceroute reports delay for each hop on the probed path by gradually increasing the time-to-live (TTL) field of probing packets. Its refinement, pathchar, returns hop-by-hop capacities, delays, and loss rates. A later advancement, NCS, also returns available capacities on each link. While providing fine-grained information, the above tools require that Internet Control Message Protocol (ICMP) be supported at each internal node. Even then, they suffer inaccuracies caused by asymmetry in routes and different priorities of ICMP and data packets. Moreover, these tools can generate a large number of probing packets, causing extra load and potential congestion. In risk-sensitive applications, security policies may even block hop-by-hop measurements.

Alternatively, the end-to-end approaches provide a solution that does not rely on the cooperation of internal network elements or the equal treatment of control/data packets. They rely on end-to-end performance metrics (e.g., end-to-end delays) experienced by data packets to solve for the corresponding hop-by-hop metrics using *network tomography*. Network tomography [4, 5] refers to the methodology of inferring internal network characteristics through end-to-end measurements. Without requiring special cooperation from internal nodes, network tomography can utilize measurements from data packets to obtain path-level information [6, 7], thus reducing the need for active probes.

A case of particular interest in network tomography is the inference of *additive* link metrics, i.e., the end-to-end metric over a path of multiple links is the sum of individual link metrics. For instance, delays are additive, while a multiplicative met-

ric (e.g., packet delivery ratio) can be expressed in an additive form using the $\log(\cdot)$ function. For additive link metrics, we can model the problem as that of solving a system of linear equations, where the unknown variables are the link metrics, and the known metrics are the end-to-end path measurements, each equal to the sum of the corresponding link metrics along a path. Thus, network tomography essentially solves this linear system of equations. In essence, network tomography provides an efficient and light-weight solution for inferring unknown link metrics from easily measurable end-to-end paths. Forasmuch as network tomography is a significant technique in revealing network internal link states, in this thesis, we focus on exploring key issues in identifying additive link metrics via end-to-end path measurements.

1.2 Research Challenges and Objectives

1.2.1 Challenges

Past experience shows that it is frequently impossible to uniquely identify all link metrics from path measurements [8, 9, 10]. For example, if two links (not necessarily adjacent) always appear together in measurement paths, then we can at most identify their sum metric but not the individual metrics. Generally, many measurement paths are *linearly dependent* in that some paths are linear combinations of the rest, and hence their measurements do not provide new information. From the perspective of linear algebra, link metrics are uniquely identifiable if and only if the number of *linearly independent* measurement paths equals the number of links. There is, however, a lack of basic understanding of the topological conditions that ensure identifiability, even in the simplified scenario of constant link metrics. Moreover, even if a network is known to be identifiable, it is still a challenge to determine a minimum set [11] of paths to measure as the total number of candidate paths can be exponential in the number of links. In addition, in cases where networks are not completely identifi-

able due to improper monitor placement or insufficient monitor deployment budget, an open question is: which sub-set of link metrics can be identified through a partial deployment of monitors, and where should the monitors be placed to identify the largest sub-set of link metrics? Therefore, it is also essential to quantitatively characterize network identifiability when the network is not completely identifiable.

1.2.2 Objectives

In this thesis, the objectives are to address the above issues by solving the challenging problems below:

- (i) Under what conditions can one uniquely identify all link metrics from end-to-end path measurements;
- (ii) Given an identifiable network, how can one construct a sufficient number of linearly independent paths and compute individual link metrics efficiently using these measured paths;
- (iii) With deployed monitors in a network, which links are identifiable from end-to-end path measurements between monitors;
- (iv) Given a network and κ monitors, where should these monitors be placed in order to identify the maximum number of links?

We study all above problems by establishing fundamental network conditions and efficient algorithms. The significance of this research is to provide insights and guidance for building effective and reliable monitoring infrastructures in real networks using network tomography.

1.3 Summary of Contributions

We study here for the *first* time the fundamental topological conditions for identifying additive link metrics using end-to-end measurements on cycle-free paths. Our

contributions are nine-fold:

1) We prove that it is generally impossible to identify all link metrics using only two monitors, irrespective of the network topology and the placement of monitors.

2) We establish necessary and sufficient conditions for identifying the metrics of all *interior links* (links not incident to any monitor) using two monitors: the network graph is (i) 2-edge-connected after removing any interior link and (ii) 3-vertex-connected after adding a direct link between the two monitors. These conditions are shown to be verifiable in $O(|L|(|V| + |L|))$ time, where $|V|$ is the number of nodes and $|L|$ is the number of links.

3) We transform the above result into a necessary and sufficient condition for identifying all link metrics using κ ($\kappa \geq 3$) monitors by embedding the network graph in an extended graph, with two virtual monitors connected to all real monitors. This condition can be verified in $O(|V| + |L|)$ time.

4) We develop an efficient algorithm to construct $|L|$ linearly independent, cycle-free paths between monitors in an arbitrary identifiable network in $O(|V| \cdot |L|)$ time, significantly improving on the exponential complexity of brute-force methods.

5) Given measurements along the constructed paths in 4), we develop an algorithm to calculate the $|L|$ link metrics in $O(|V| + |L|)$ time, improving on the $O(|L|^3)$ complexity of the existing solution [12]. The algorithm exploits a nested structure of the constructed paths to avoid the need for explicitly inverting the measurement matrix.

6) We develop an algorithm (called DAIL) to determine all identifiable links for a given placement of monitors. DAIL has a complexity of $O(|V| + |L|)$.

7) We propose an algorithm to place monitors in an arbitrary network in $O(|V| + |L|)$ time, which guarantees the identifiability of all link metrics with the minimum number of monitors. Our evaluations on both randomly-generated and real topologies show that the proposed algorithm requires a substantially smaller number of

monitors than a baseline solution.

8) To maximize the number of identifiable links using κ monitors (κ monitors are not sufficient to identify the entire network), we first notice that only a subset of nodes can be involved in an optimal placement, based on which we develop a preprocessing algorithm to select a set of candidate monitors $\mathcal{S}$, guaranteed to contain an optimal κ -monitor placement, whose size is proportional to the number of triconnected components and usually much smaller than $|V|$. We then develop a polynomial-time algorithm that incrementally selects κ monitors from $\mathcal{S}$ such that each selected monitor maximizes the additional number of identifiable links.

9) The proposed monitor placement algorithm in 8) is optimal for 2-vertex-connected networks. For non-2-vertex-connected networks, we show via numerical evaluations on several real ISP topologies that our algorithm closely approximates the optimal number of identifiable links for all considered values of κ , while incurring a much lower complexity than an exhaustive search.

1.4 Thesis Organization

The organization of the rest of the thesis is as follows.

In Chapter 2, we provide the literature review and problem formulation for additive link metric identification. Chapter 3 presents a set of easily verifiable network topological conditions. In regards to completely identifiable networks, Chapter 4 addresses efficient algorithms for path construction and link identification. Under given monitor locations, algorithms for quantifying network identifiability in terms of the number of identifiable links are established in Chapter 5. Next, we propose optimal monitor placement algorithms and heuristics to maximize network identifiability in Chapter 6. Finally, Chapter 7 concludes the thesis and identifies future work.

Literature Review and Problem Formulation in Network Tomography

2.1 Literature Review

2.1.1 Complete Network Identifiability

Based on the model of link metrics, existing work for identifying link metrics can be broadly classified into statistical and algebraic approaches. Statistical approaches model link metrics as random variables with (partially) unknown probability distributions, and apply various parametric/nonparametric techniques to estimate the link metric distributions from realizations of path metrics [13, 1, 14]. Algebraic approaches consider link metrics as unknown constants, and use linear algebraic techniques to compute link metrics from cumulative path metrics [8, 9]. Both approaches emphasize inferring as much about link metrics as possible from available path measurements. However, the important question of whether the path measurements contain sufficient information to uniquely identify the underlying link metrics is often ignored.

With link metrics modeled as random variables, multicast, if supported, can be exploited as a measurement method with broad coverage and low overhead [15, 16, 17]. Sub-trees and unicast are employed in [7, 18] as alternatives, due to the inflexibility of multicasting to all receivers. Employing multicast, [7, 19] derive the

necessary and sufficient conditions on the multicast tree for identifying all link metric distributions. A novel approach proposed in [10] employs the Fourier transform of the observable path metric distributions to estimate the unobservable link metric distributions. All the above methods implicitly assume the links to be identifiable, and the multicast-based methods require multiple monitors to participate in the measurement process. In contrast, we assume unicast measurements in this thesis and focus on establishing topological conditions for identifying all link metrics by using the minimum number of monitors.

For constant link metrics, [9, 20] show that it is challenging to solve the inverse problem due to the presence of linearly dependent paths. If all link metrics are *binary* (normal/failed), [21] proves that the network must be $(k + 2)$ -edge-connected to identify up to k failed links by using one monitor measuring cycles. If the link metric of interest is the bandwidth, [22] proposes an end-to-end probing technique to locate the bottleneck link in real Internet. However, for arbitrary valued additive link metrics, few positive results are known. If the network is directed (links have different metrics in different directions), [14] proves that not all link metrics are identifiable unless every non-isolated node is a monitor. Even if every node is a monitor, unique link identification is still impossible if measurement paths are constrained to cycles [8]. If the network is undirected (links have equal metrics in both directions), RRT (Round Trip Time) is exploited in [23, 2, 24, 25] to infer link metrics, with the assumption of ICMP (Internet Control Message Protocol) being supported in all inner network elements. However, ICMP-enabled network is not widely employed and the inferred link information is inaccurate due to routing protocols and the low priority of ICMP packets in IP networks. As a fundamental work, [26] derives the first necessary and sufficient conditions on the network topology for identifying all link metrics, given that monitors can measure cycles or paths possibly containing cycles. A similar study in [27] characterizes the minimum number of measurements need-

ed to identify a broader set of link metrics (including both additive and nonadditive metrics), under the stronger assumption that measurement paths can contain repeated links. Since routing along cycles is typically prohibited in real networks, it remains open as to what the conditions are if only cycle-free paths can be measured. In this regard, we investigate the fundamental relationships between additive link identifiability, network topology, and the number/placement of monitors in this thesis.

2.1.2 Path Construction and Link Identification

Once link identifiability is confirmed, a follow-up problem is to construct linearly independent paths between monitors to perform measurements. For multicast-based statistical approaches [15, 16, 7], measurement paths can be conducted along the multicast tree. For the scenario with link metrics being modeled as constants, [26] proposes an efficient algorithm to construct measurement paths, given that monitors can measure cycles or paths containing cycles. Given measurements on $|L|$ linearly independent paths ($|L|$ is the number of links), a general solution for computing link metrics using classic linear system solvers (e.g. Gaussian elimination with pivoting) takes $O(|L|^3)$ time [12], which can be slow for large $|L|$. In this thesis, we overcome this drawback by developing efficient algorithms for path construction and link identification in arbitrary identifiable networks, employing only cycle-free paths.

2.1.3 Partial Network Identifiability

Imposing more constraints on the network, [13] proposes algorithms to identify the worst performing links, with the assumption that most links do not exhibit severe losses or delays. Similarly, when all but k link metrics are *zero*, compressive sensing techniques are used to identify the k non-zero link metrics [28, 29].

However, little is known for characterizing partial network identifiability in cases of general network conditions. An algorithm is developed in [30] to determine the

maximum number of linearly independent cycles (i.e., the maximum rank of measurement matrix, see (2.1) in Section 2.2) and the corresponding identifiable links with a single monitor. Constraining all measurement paths to be cycle-free, [31] establishes an upper bound for the rank of the measurement matrix in both directed and undirected graphs. However, this rank is defined *over the Galois Field* [32] $GF(2)$, and how this rank is related to the identifiable links in the given network remains unknown. Employing the property that some measurement paths are linearly dependent, [33, 34] propose linear algebraic solutions to compute the unmeasured sub-paths based on all available path measurements. To infer link weights under given routing protocols, [35] proposes algorithms for approximating the relative value of some link metrics. Under arbitrarily controllable packet routing, we build an efficient algorithm in [36] to determine all identifiable links in a network with only two monitors. Breaking the two-monitor constraint, in this thesis, we establish a linear-time algorithm to determine all identifiable links in an arbitrary network with any monitor deployment.

2.1.4 Monitor Placement

It is generally challenging [25, 37, 38] to place monitors for ensuring complete network identifiability. For the issue of link failure localization, it is provable [39] that the problem of monitor placement is NP-hard. In [25, 37], the problem of placing the minimum number of monitors to identify all link metrics (with arbitrary values) under *uncontrollable* routing is also proved to be NP-hard, and the NP-hardness persists even if some nodes can control their local routing policy [40]. In contrast, we show in this thesis that under *controllable* cycle-free routing, the problem of minimum monitor placement for complete identifiability can be solved optimally in linear time. However, in cases where we cannot afford even the minimum number of monitors required for complete identification, it remains open how a limited number of

monitors affects network identifiability. We address this problem by presenting an efficient monitor placement algorithm to maximize network identifiability.

2.2 Problem Formulation

In this thesis, all investigated problems are in the context of *controllable, cycle-free* measurements, i.e., monitors can direct measurement packets to selected paths as long as they do not contain cycles. Such routing is generally supported in networks under common administration (e.g., single-ISP networks), or overlay networks formed by overlay nodes and physical paths between these nodes (modeled as overlay links), where controllable routing can be achieved by enabling strict (or loose) source routing (an IP option) [41] at all network nodes during measurement. More importantly, the assumption of controllable, cycle-free measurements captures capabilities of a new generation of networks performing *Software-Defined Networking (SDN)* [42, 43, 44]. In SDN, routing is performed on a per-flow basis in two steps: a route setup phase where a (logically) centralized SDN controller specifies the path of a flow by setting entries in the forwarding tables of switches along the path, and a data forwarding phase where data of the flow are forwarded hop by hop by switches according to their forwarding tables. Therefore, monitors, in cooperation with the SDN controller, can dictate paths of measurement packets during route setup, while the cycle-free constraint precludes formation of endless cycles in the data forwarding phase. Moreover, in SDN, there exists a communication bottleneck that lies in the control channel between the SDN controller and all network nodes; therefore, directly measuring each link metric initiated by the controller can easily congest this control channel especially in large networks. In this regard, direct performance measurement is rather impractical even under the network settings of SDN. Hence, network tomography can be used to assist in accurately monitoring network perfor-

mance experienced by data traffic while reducing measurement overhead.

In this thesis, all link metrics are assumed to be additive and constant. Our “constant” link metric refers to one that either changes slowly relative to the measurement process, or that is a statistical characteristic (e.g., mean, variance) of the link that remains constant over time¹. We assume the network topology is known² and model it as an undirected graph³ $\mathcal{G} = (V, L)$, where V and L are the sets of nodes and links, respectively. Note that graph $\mathcal{G}$ can represent a logical topology where each link in $\mathcal{G}$ corresponds to a combination of physical links/nodes, e.g., in an overlay network. Without loss of generality, we assume $\mathcal{G}$ is connected, as different connected components have to be monitored separately. Denote the link incident to nodes u and v by uv ; links uv and vu are assumed to have the same metric. Certain nodes in V are monitors and can initiate/collect measurements. We assume that each link in $\mathcal{G}$ has two distinct end-points (i.e., no self-loop), and there is at most one link connecting a pair of nodes. Last, no link metrics in $\mathcal{G}$ are known prior to taking measurements. All graph theory notations used in this thesis are summarized in the table of Mathematical Notations (following the convention of [47]). Throughout this thesis, an area with a dashed border in a graph denotes a sub-graph (nodes/links on the dashed border are also part of the sub-graph), and a solid line denotes a link/path/cycle.

Let $n := |L|$ denote the number of links in $\mathcal{G}$, $\{l_i\}_{i=1}^n$ the set of links in $\mathcal{G}$, $\mathbf{w} = (W_{l_1}, \dots, W_{l_n})^T$ the column vector of all link metrics, and $\mathbf{c} = (W_{\mathcal{P}_1}, \dots, W_{\mathcal{P}_\gamma})^T$ the column vector of all available path measurements, where γ is the number of measurement paths and $W_{\mathcal{P}_i}$ is the sum of link metrics along measurement path $\mathcal{P}_i$. $W_{\mathcal{P}_i}$ can be obtained explicitly by probing path $\mathcal{P}_i$ or extracted implicitly from da-

¹In this case, end-to-end measurements are also statistical characteristics, e.g., path mean/variance. In the case of variance, we also need the independence between link qualities to make the metric additive.

²Network topology can be obtained through exchanging control packets [45, 46] explicitly. Note that network internal states cannot be inferred from these control packets in an accurate manner due to the unequal treatment of control and data packets.

³We use the terms *network* and *graph* interchangeably.

ta packets that traverse $\mathcal{P}_i$. Since we assume that monitors can control the routing of measurement packets as long as the path starts and ends at distinct monitors and does not contain repeated nodes, in the language of graph theory, we limit measurements to *simple paths* (in contrast, a *non-simple path* may contain repeated nodes). Then the path measurements can be expressed in terms of the unknown link metrics through the following linear system:

$$\mathbf{R}\mathbf{w} = \mathbf{c}, \quad (2.1)$$

where $\mathbf{R} = (R_{ij})$ is a $\gamma \times n$ *measurement matrix*, with each entry $R_{ij} \in \{0, 1\}$ denoting whether link l_j is present on path $\mathcal{P}_i$. The network tomography problem therefore is to invert this linear system to solve for $\mathbf{w}$ given $\mathbf{R}$ and $\mathbf{c}$.

A link is *identifiable* if the associated link metric can be uniquely determined from path measurements; network $\mathcal{G}$ is *completely identifiable* if all links in $\mathcal{G}$ are identifiable. Otherwise, the link or the network is said to be *unidentifiable*. Given the above linear system, $\mathcal{G}$ is completely identifiable if and only if $\mathbf{R}$ in (2.1) has full column rank, i.e., $\text{rank}(\mathbf{R}) = n$. In other words, to uniquely determine $\mathbf{w}$, there must be n linearly independent simple paths between monitors. However, as illustrated in Section 2.2.2, even if $\text{rank}(\mathbf{R}) < n$, it may still be possible to identify some of the link metrics, i.e., $\mathcal{G}$ can be *partially identifiable*. The number of identifiable links therefore provides a *quantitative* measure of network identifiability.

2.2.1 Research Issues

As discussed in Section 1.2.2, four research issues are investigated in this thesis: (i) derive necessary and sufficient conditions for identifying all link metrics in network $\mathcal{G}$ (or certain sub-graphs of $\mathcal{G}$) under a given placement of κ ($\kappa \geq 2$) monitors by solving the linear system (2.1); (ii) develop efficient algorithms for constructing monitor-to-monitor measurement paths and calculating each link metric in complete-

ly identifiable $\mathcal{G}$; (iii) determine the set of links in $\mathcal{G}$ that are identifiable for a given monitor placement, and (iv) find an optimal placement of a given number of monitors to maximize the number of identifiable links.

Although the linear algebraic answer to the first research issue is straightforward (the number of linearly independent paths equals the number of links), a useful answer should be expressed in terms of externally verifiable network properties such as the network topology and the number/placement of monitors. To this end, we establish necessary and sufficient conditions on network topology and monitor placement for uniquely identifying link metrics from controllable, cycle-free measurements between the monitors. To determine the metrics of all identifiable links in the second research issue, although one can enumerate all possible measurement paths until finding n linearly independent ones, such a method will incur a large cost (exponential in n) and is thus unsuitable for large networks. A smart strategy for finding n linearly independent paths will not only save resources in collecting measurements, but also increase the efficiency of calculating link metrics by discarding redundant linear equations. For the third and fourth research issues, they may be answered by finding the null space of measurement matrix $\mathbf{R}$ and enumerating all possible monitor placements, the complexity of such approaches will explode exponentially as the network size grows. We take a different approach here by drawing insights from established identifiability conditions in Chapter 3 that relate link identifiability to network topology and monitor placements. These insights then form the basis of our efficient polynomial-time algorithms addressing the third and fourth research issues.

We note that our goal is to conduct fundamental research in characterizing identifiable scenarios for constant additive link metrics under the arbitrarily controllable cycle-free routing mechanism. In cases where the link metrics are modeled as random variables, additional measurements [48] or preprocessing of the measurement matrix [49] may assist in revealing more network internal information. However,

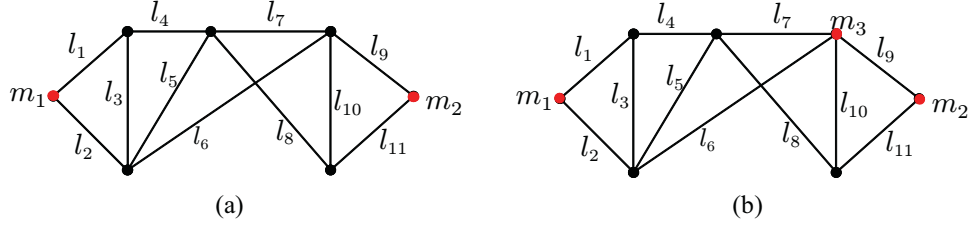


Figure 2.1: Sample network with given monitor placement: (a) two monitors m_1 and m_2 ; (b) three monitors m_1 , m_2 , and m_3 .

estimating link parameters under statistical models or identifying non-additive link metrics (i.e., bandwidth) are beyond the scope of this thesis.

2.2.2 Illustrative Example

Fig. 2.1 (a) displays a sample network with two monitors $\{m_1, m_2\}$, and eleven links (links l_1 – l_{11}). To identify these links, ten $m_1 \rightarrow m_2$ simple paths are constructed to form the measurement matrix $\mathbf{R}$:

$$\begin{aligned}
 \mathcal{P}_1 &: l_1 \ l_4 \ l_8 \ l_{11} \\
 \mathcal{P}_2 &: l_1 \ l_4 \ l_7 \ l_9 \\
 \mathcal{P}_3 &: l_2 \ l_6 \ l_9 \\
 \mathcal{P}_4 &: l_2 \ l_3 \ l_4 \ l_7 \ l_9 \\
 \mathcal{P}_5 &: l_2 \ l_5 \ l_8 \ l_{10} \ l_9 \\
 \mathcal{P}_6 &: l_2 \ l_5 \ l_7 \ l_9 \\
 \mathcal{P}_7 &: l_1 \ l_3 \ l_6 \ l_9 \\
 \mathcal{P}_8 &: l_1 \ l_4 \ l_5 \ l_6 \ l_9 \\
 \mathcal{P}_9 &: l_1 \ l_4 \ l_7 \ l_{10} \ l_{11} \\
 \mathcal{P}_{10} &: l_1 \ l_3 \ l_6 \ l_7 \ l_8 \ l_{11}
 \end{aligned}
 \Rightarrow \mathbf{R} = \begin{pmatrix}
 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1
 \end{pmatrix}. \quad (2.2)$$

where $R_{ij} = 1$ if and only if link l_j is on path $\mathcal{P}_i$. Measuring the performance along these paths provides a set of linear equations $\mathbf{R}\mathbf{w} = \mathbf{c}$, where $\mathbf{c}$ is the vector of path metrics measured at m_2 . In Fig. 2.1 (a), it can be verified that adding any other simple path between m_1 and m_2 does not provide further information about the link metrics since it is linearly dependent with the paths in (2.2). In this example, $\mathbf{R}$ is *non-invertible*, and thus the metrics in $\mathbf{w}$ cannot be completely identified. Nevertheless, measurements of the paths in (2.2) provide sufficient information to identify the

metrics of seven links: $l_3, l_4, l_5, l_6, l_7, l_8$, and l_{10} . Specifically, $W_{l_3} = (W_{\mathcal{P}_4} + W_{\mathcal{P}_7} - W_{\mathcal{P}_2} - W_{\mathcal{P}_3})/2$, $W_{l_4} = (W_{\mathcal{P}_4} + W_{\mathcal{P}_8} - W_{\mathcal{P}_6} - W_{\mathcal{P}_7})/2$, $W_{l_5} = (W_{\mathcal{P}_6} + W_{\mathcal{P}_8} - W_{\mathcal{P}_2} - W_{\mathcal{P}_3})/2$, $W_{l_6} = (W_{\mathcal{P}_3} + W_{\mathcal{P}_8} + W_{\mathcal{P}_{10}} - W_{\mathcal{P}_1} - W_{\mathcal{P}_6} - W_{\mathcal{P}_7})/2$, $W_{l_7} = (W_{\mathcal{P}_2} + W_{\mathcal{P}_{10}} - W_{\mathcal{P}_1} - W_{\mathcal{P}_7})/2$, $W_{l_8} = (W_{\mathcal{P}_2} + W_{\mathcal{P}_5} + W_{\mathcal{P}_{10}} - W_{\mathcal{P}_6} - W_{\mathcal{P}_7} - W_{\mathcal{P}_9})/2$, and $W_{l_{10}} = (W_{\mathcal{P}_5} + W_{\mathcal{P}_9} - W_{\mathcal{P}_1} - W_{\mathcal{P}_6})/2$. Moreover, it can be verified that if we change the locations of these two monitors, then the number of identifiable links will decrease. Therefore, the monitor placement in Fig. 2.1 (a) is optimal in that it maximizes the number of identifiable links among all 2-monitor placements. Note that a path such as $l_2 l_5 l_4 l_3 l_6 l_9$ between m_1 and m_2 cannot be measured because it contains a cycle.

Nevertheless, if we add a third monitor (m_3) to the network in Fig. 2.1 (a), displayed in Fig. 2.1 (b), then eleven end-to-end linearly independent paths (one $m_1 \rightarrow m_2$ path, seven $m_1 \rightarrow m_3$ paths and three $m_3 \rightarrow m_2$ paths) can be constructed to form the measurement matrix $\mathbf{R}$:

$$\begin{aligned}
 & m_1 \rightarrow m_2 : l_1 l_4 l_8 l_{11} \\
 & m_1 \rightarrow m_3 : l_1 l_4 l_7 \\
 & \quad l_2 l_6 \\
 & \quad l_2 l_3 l_4 l_7 \\
 & \quad l_2 l_5 l_8 l_{10} \\
 & \quad l_2 l_5 l_7 \\
 & \quad l_1 l_3 l_6 \\
 & \quad l_1 l_4 l_5 l_6 \\
 & m_3 \rightarrow m_2 : l_9 \\
 & \quad l_{10} l_{11} \\
 & \quad l_6 l_5 l_8 l_{11}
 \end{aligned}
 \Rightarrow \mathbf{R} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad (2.3)$$

In this case, $\mathbf{R}$ in (2.3) is invertible, and thus $\mathbf{w}$ can be uniquely identified, i.e., $\mathbf{w} = \mathbf{R}^{-1}\mathbf{c}$. In Fig. 2.1 (b), other simple paths can be measured as well, although they do not provide further information since the measurement matrix already reaches full rank. In Chapter 6, we will algorithmically prove that three monitors are the minimum requirement for completely identifying the network in Fig. 2.1.

Network Identifiability Conditions

In this chapter, we built a set of necessary and sufficient conditions for network identifiability that are explicitly expressed in terms of network topology and the number/placement of monitors. In a network containing only two monitors, we first present a negative result that no matter where we place the monitors, we cannot identify all link metrics using only two monitors; nevertheless, interior links not incident to any monitor might be identifiable. Accordingly, the second main result in this chapter is the necessary and sufficient conditions for identifying all the interior links using two monitors. Finally, we extend these conditions to investigate on complete network identification using three or more monitors.

3.1 Unidentifiability with Two Monitors

Since at least *two* monitors are required to identify link metrics by monitoring simple paths, in this section, we investigate if two monitors suffice to identify all link metrics in the network. Suppose that two distinct nodes are selected to serve as monitors. Each measurement starts at one monitor and terminates at the other via a controllable (i.e., predetermined) simple path. The termination node then reports the end-to-end metric, which becomes an entry in the measurement vector $\mathbf{c}$. From the perspective of graph theory, such a network can be represented as $\mathcal{G} = (\{m_1, m_2, v_0, \dots, v_k\}, L)$, where m_1 and m_2 are the monitors, $\{v_0, \dots, v_k\}$ are the non-monitors, and $|L| = n$. Let $m_1 m_2$ be a direct link between m_1 and m_2 (if it exists), $L(\mathcal{G})$ the set of links

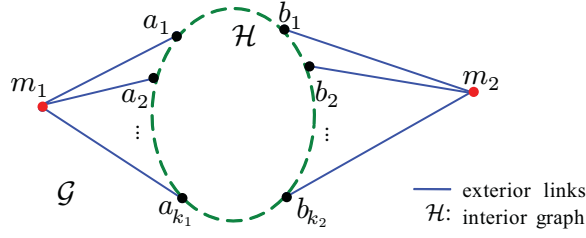


Figure 3.1: Organizing graph $\mathcal{G}$ into exterior links and interior graph ($\{a_i\}_{i=1}^{k_1}$ and $\{b_j\}_{j=1}^{k_2}$ may have overlap).

in $\mathcal{G}$, and $L(v)$ the set of links incident to node v (see the table of Mathematical Notations). Since $m_1 m_2$ can be easily identified through a one-hop measurement, we assume without loss of generality that $m_1 m_2 \notin L(\mathcal{G})$ (i.e., there is no direct link) in Sections 3.1 and 3.2, where only two monitors are used.

Definition 1.

- a) The *interior graph* $\mathcal{H}$ of $\mathcal{G}$ is the sub-graph obtained by removing the monitors and their incident links, i.e., $\mathcal{H} := (V \setminus M, L \setminus L_M)$ for $M = \{m_1, m_2\}$ and $L_M = L(m_1) \cup L(m_2)$.
- b) We refer to links incident to monitors, i.e., $L(m_1) \cup L(m_2)$, as *exterior links*, and the remaining links as *interior links*.

Theorem 3.1. For any given network topology $\mathcal{G}$ with $n \geq 2$ (n is the number of links), $\mathcal{G}$ is not completely identifiable with two monitors, irrespective of their placement.

Proof. Any $\mathcal{G}$ with $||\mathcal{G}|| \geq 2$ can be organized as in¹ Fig. 3.1. Let $N(v)$ denote the set of neighboring nodes of node v . We define $A := \{a_1, a_2, \dots, a_{k_1}\} = N(m_1)$ and $B := \{b_1, b_2, \dots, b_{k_2}\} = N(m_2)$ to be the sets of neighbors of m_1 and m_2 , respectively, where $k_1 := |A|$, $k_2 := |B|$ and A, B can overlap ($m_1, m_2 \notin A \cup B$).

¹If certain links in $\mathcal{G}$ cannot be included in any possible paths constructed from m_1 to m_2 in Fig. 3.1, then these links are unidentifiable, resulting in a disconnected or one-edge-connected interior graph $\mathcal{H}$. In this thesis, an area with a dashed border denotes a sub-graph (nodes/links on the dashed border are also part of the sub-graph, e.g., $\{a_i\}_{i=1}^{k_1}$ and $\{b_j\}_{j=1}^{k_2}$ in Fig. 3.1 are part of $\mathcal{H}$), and a solid line denotes a link/path/cycle.

Assuming that $\mathcal{H}$ is connected and all link metrics in $\mathcal{H}$ are known, we can reduce any equation associated with a simple path $\mathcal{P}$ between m_1 and m_2 to the form:

$$W_{m_1 a_i} + W_{b_j m_2} = \phi_{ij}, \quad (3.1)$$

for some $a_i \in A$ and $b_j \in B$. This is obtained by rewriting the original equation $W_{m_1 a_i} + W_{\mathcal{P}'_{ij}} + W_{b_j m_2} = W_{\mathcal{P}_{ij}}$ ($\mathcal{P}'_{ij}$ is the segment of $\mathcal{P}_{ij}$ in $\mathcal{H}$) to place the unknowns on the left-hand side, and setting $\phi_{ij} := W_{\mathcal{P}_{ij}} - W_{\mathcal{P}'_{ij}}$. Thus, we obtain $k_1 \times k_2$ equations from all of the simple paths between m_1 and m_2 , each corresponding to the sum of the metrics of one link incident to m_1 and one link incident to m_2 . The corresponding reduced measurement matrix is (each column corresponding to an unknown link metric):

$$\mathbf{R} = \begin{array}{c} \begin{array}{cc} W_{m_1 a_1} \cdots W_{m_1 a_{k_1}} & W_{b_1 m_2} \cdots W_{b_{k_2} m_2} \end{array} \\ \left(\begin{array}{cc|cc} 1 & & 1 & \\ & & & 1 \\ & & & \ddots \\ & & & & 1 \\ \hline & 1 & 1 & \\ & 1 & & 1 \\ & \vdots & & \ddots \\ & 1 & & & 1 \\ \hline & & \ddots & & \vdots \\ \hline & & & 1 & 1 \\ & & & 1 & \\ & & & \vdots & \ddots \\ & & & 1 & & 1 \end{array} \right), \quad (3.2)$$

where the blank entries are zero. We apply the following linear transformations to $\mathbf{R}$. For each $q = 1, \dots, k_1 - 1$ and $i = 2, \dots, k_2$, replace $\text{row}(qk_2 + i)$ by $\text{row}(qk_2 + i) - \text{row}(i) - \text{row}(qk_2 + 1) + \text{row}(1)$; it can be verified that the result is

a row of zeros. Ignoring rows of zeros, $\mathbf{R}$ transforms into

$$\mathbf{R}' = \left(\begin{array}{c|c} W_{m_1 a_1} \cdots W_{m_1 a_{k_1}} & W_{b_1 m_2} \cdots W_{b_{k_2} m_2} \\ \hline \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} & \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \end{array} \right) \left. \begin{array}{l} \left. \begin{array}{l} k_2 \\ \text{rows} \end{array} \right\} \\ \left. \begin{array}{l} k_1 - 1 \\ \text{rows} \end{array} \right\} \end{array} \right\}, \quad (3.3)$$

where the rows are linearly independent, and the number of rows equals $k_1 + k_2 - 1$. This transformation preserves the rank of $\mathbf{R}$ because $\mathbf{R}$ can be reconstructed from $\mathbf{R}'$ as follows: the first k_2 rows in $\mathbf{R}$ and $\mathbf{R}'$ are identical; row($qk_2 + 1$) in $\mathbf{R}$ equals row($k_2 + q$) in $\mathbf{R}'$ ($q = 1, \dots, k_1 - 1$); row($qk_2 + i$) in $\mathbf{R}$ equals row(i) + row($k_2 + q$) - row(1) in $\mathbf{R}'$ ($q = 1, \dots, k_1 - 1, i = 2, \dots, k_2$). Hence, the rank of $\mathbf{R}$ equals $k_1 + k_2 - 1$. When $\mathcal{H}$ is not connected, some rows in $\mathbf{R}$ may not exist because there is no simple path connecting the corresponding nodes in A and B , and the rank of $\mathbf{R}$ may be even smaller. Since there are $k_1 + k_2$ unknown variables $(W_{m_1 a_i})_{i=1}^{k_1}$ and $(W_{b_j m_2})_{j=1}^{k_2}$, they cannot be uniquely determined even if all link metrics in $\mathcal{H}$ are already known. Therefore, $\mathcal{G}$ with $||\mathcal{G}|| \geq 2$ is unidentifiable using two monitors. $\square$

In fact, we can show that none of the exterior links is identifiable because metric of any one exterior link will add a linearly independent row to $\mathbf{R}'$ in (3.3), thus yielding full column rank of $\mathbf{R}'$ (i.e., metrics of the rest exterior links are also determinable).

Corollary 3.2. None of the exterior links (except $m_1 m_2$) can be identified with two monitors.

Proof. Assume that all the interior link metrics are known. We prove Corollary 3.2 by contradiction. From the proof of Theorem 3.1, we see that the transformed measurement matrix $\mathbf{R}'$ in (3.3) gives a maximum set of linearly independent equations (one equation per row) regarding the exterior link metrics $\{W_{m_1 a_i}\}_{i=1}^{k_1}$ and $\{W_{b_j m_2}\}_{j=1}^{k_2}$. Suppose there exists one exterior link metric, say $W_{m_1 a_i}$, which is identifiable. Since each row in $\mathbf{R}'$ only involves two exterior link metrics, it is easy to see that knowing the metric of any exterior link (i.e., $W_{m_1 a_i}$ according to the assumption) will allow unique identification of all the other exterior link metrics, contradicting the fact that $\mathbf{R}'$ is rank deficient and hence not able to identify all the metrics. Note the direct link $m_1 m_2$ (if exists) is always identifiable. Therefore, none of the exterior links (except $m_1 m_2$) can be identified with two monitors. $\square$

Discussions on Paths with Cycles: At the end of Section III-B in [26], the authors raise the question whether or not monitoring non-simple paths (i.e., paths that may contain cycles) between two monitors suffices to identify all link metrics in the network². According to Corollary 3.2, the exterior links cannot be identified even if all the interior link metrics are known; allowing cycles in the interior graph $\mathcal{H}$ provides no additional information regarding the exterior links. Consequently, the answer to that question in [26] is that monitoring (simple or non-simple) paths between two monitors is *not* sufficient to identify all link metrics.

3.2 Identifiability of Interior Links with Two Monitors

Section 3.1 shows that we can never identify all link metrics using only two monitors no matter where these two monitors are placed. Nevertheless, network administra-

²We restrict the paths from containing repeated monitors to exclude measurement of cycles, as is already solved in [26].

tors are more interested in using end-to-end measurements to infer the qualities of links that are at least *one-hop* away. Therefore, in this section, we only focus on the interior graph $\mathcal{H}$ and derive necessary and sufficient conditions on the network topology $\mathcal{G}$ for identifying all links in $\mathcal{H}$ using two monitors (m_1 and m_2), under the earlier assumption that no link metrics (including those of links incident to m_1 and m_2) are known ahead of time.

Before going into details, we first point out that it is sufficient to solve the case in which $\mathcal{H}$ is a connected graph. This is because if $\mathcal{H}$ consists of K_H ($K_H \geq 2$) connected components $\mathcal{H}_i$ ($i = 1, \dots, K_H$), we can decompose the entire graph $\mathcal{G}$ into sub-graphs $\mathcal{G}_i := \mathcal{H}_i + m_1 + m_2$, with $\mathcal{G} = \cup_{i=1}^{K_H} \mathcal{G}_i$ (see the definition of graph union in Mathematical Notations). Since none of the $m_1 \rightarrow m_2$ simple paths in $\mathcal{G}_i$ can traverse $\mathcal{G}_j$ ($i \neq j$), the identification of links within different $\mathcal{G}_i$'s is mutually independent. Therefore, in the rest of this section, we assume $\mathcal{H}$ is connected with $|\mathcal{H}| \geq 1$. Our result can be applied to each $\mathcal{G}_i$ separately when $\mathcal{H}$ is disconnected.

We show that the exterior links can never be identified using two monitors (see Corollary 3.2), but the interior links may be identifiable (see the example in Section 2.2.2). To characterize the identifiability of the interior graph of a network with two monitors, we use the following notions from graph theory.

Definition 2. Graph $\mathcal{G}$ of $|V|$ vertices ($|L|$ edges) is said to be k -vertex-connected (k -edge-connected) if $k \leq |V| - 1$ ($k \leq |L| - 1$) and deleting any subset of up to $k - 1$ vertices ($k - 1$ edges) does not disconnect $\mathcal{G}$.

The benefit of using k -vertex-connected and k -edge-connected to characterize network identifiability is that it is easily verifiable via existing algorithms (see Section 3.4) in graph theory. Therefore, with Definition 2, we are ready to present the following necessary and sufficient conditions for characterizing the identifiability of the network interior graph.

Theorem 3.3. Assume that the interior graph $\mathcal{H}$ (with $||\mathcal{H}|| \geq 1$) of $\mathcal{G}$ under a given placement of two monitors (m_1 and m_2) is connected and direct link m_1m_2 (incident to m_1 and m_2) does not exist in $L(\mathcal{G})$. The necessary and sufficient conditions for identifying all link metrics in $\mathcal{H}$ are:

- ① $\mathcal{G} - l$ is 2-edge-connected for every interior link l in $\mathcal{H}$;
- ② $\mathcal{G} + m_1m_2$ is 3-vertex-connected.

The necessity proof of Theorem 3.3 can be done by contradiction, i.e., if Conditions ①–② in Theorem 3.3 are not satisfied, some interior links in $\mathcal{G}$ are guaranteed to be unidentifiable. While for the sufficiency of Theorem 3.3, we prove it by construction. In particular, we observe that when Conditions ①–② in Theorem 3.3 are satisfied, there exist two special sub-graphs, each containing an identifiable link. This observation motivates us to construct one of these two special sub-graphs for each interior link. If all interior links can be identified by path measurements in their associated sub-graphs, then we can prove that all interior links are identifiable. Therefore, the key idea behind the sufficiency proof is that we first formally present the two types of identifiable links that are involved in two special sub-graphs. Then we prove that each interior link must fall into one of these two types of identifiable links when Conditions ①–② in Theorem 3.3 are satisfied, thus completing the proof. The detailed proofs of Theorem 3.3 are shown as follows.

3.2.1 Proof of Theorem 3.3: Necessary Part

Suppose all links in $\mathcal{H}$ are identifiable. We prove the necessity of Conditions ①–② in Theorem 3.3 by contradiction.

1) Let $l_1 \in L(\mathcal{H})$ be an arbitrary interior link. If $\mathcal{G} - l_1$ is disconnected, then l_1 is a bridge³ in $\mathcal{G}$ (shown in Fig. 3.2 (a)). If $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 3.2 (a) each contains

³A link whose removal will disconnect the graph is a bridge [47].

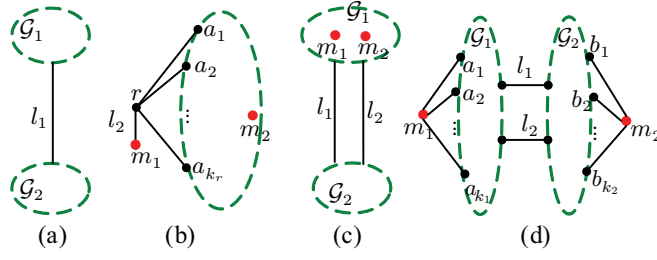


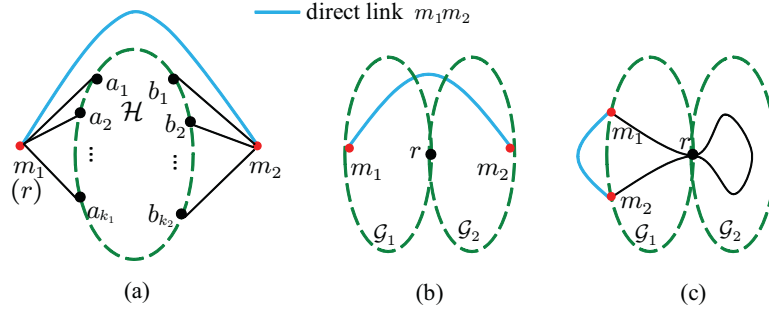
Figure 3.2: Illustration of Condition ①, where $\{l_1, l_2\}$ is an edge cut in (c) and (d).

a monitor, then l_1 is unidentifiable by Lemma D.1 (Appendix D). If m_1 and m_2 are both in $\mathcal{G}_1$ (or $\mathcal{G}_2$), then l_1 cannot be included in any measurement path (otherwise, l_1 will be used more than once), and is thus unidentifiable. Both cases contradict the assumption that all interior links are identifiable.

2) Suppose there is a bridge l_2 in $\mathcal{G} - l_1$. If l_2 is an exterior link, as shown in Fig. 3.2 (b), then by Lemma D.1 (Appendix D), its adjacent interior links $ra_i \in L(\mathcal{H})$ are unidentifiable, contradicting the assumption that all interior links are identifiable. Thus, l_2 must be an interior link. Since by 1), an interior link cannot be a bridge in $\mathcal{G}$, $\{l_1, l_2\}$ must be an edge cut as shown in Fig. 3.2 (c) and 3.2 (d). If both m_1 and m_2 are in $\mathcal{G}_1$ as in Fig. 3.2 (c), then all $m_1 \rightarrow m_2$ paths traversing l_1 must traverse l_2 as well. Thus, we can at most identify $W_{l_1} + W_{l_2}$, but not W_{l_1} and W_{l_2} individually. If m_1 is in $\mathcal{G}_1$ and m_2 is in $\mathcal{G}_2$ as in Fig. 3.2 (d), then all $m_1 \rightarrow m_2$ paths traverse either l_1 or l_2 , but not both. Assuming that $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 3.2 (d) are connected and all link metrics in them are known, then the resulting measurement matrix $\mathbf{R}_1$ is similar to (3.2), except that each row in $\mathbf{R}_1$ has a new entry associated with W_{l_1} or W_{l_2} :

$$\mathbf{R}_1 = \begin{pmatrix} \begin{matrix} \text{Exterior} \\ \text{Links} \end{matrix} & \begin{matrix} W_{l_1} & W_{l_2} \end{matrix} \\ \mathbf{R} & \left| \begin{matrix} \mathbf{e}_1 \\ \mathbf{e}_1 \end{matrix} \right| \end{pmatrix}.$$

Here blank entries correspond to zeroes, $\mathbf{R}$ is given by (3.2), and $\mathbf{e}_1$ is a $(k_1 \times k_2)$ -element column vector of all ones. Let $\mathbf{R}_2$ and $\mathbf{R}_3$ denote the first and last $k_1 \times k_2$ rows in $\mathbf{R}_1$, respectively. Then applying the same linear transformations as applied to (3.2) to both $\mathbf{R}_2$ and $\mathbf{R}_3$, we can transform $\mathbf{R}_1$ into $\mathbf{R}'_1$ (ignoring rows of zeros):

Figure 3.3: Possible scenarios of a cut-vertex r .

$$\mathbf{R}'_1 = \left(\begin{array}{c|c|c} \text{Exterior} & W_{l_1} & W_{l_2} \\ \text{Links} & & \\ \hline \mathbf{R}' & \mathbf{e}_2 & \\ \hline \mathbf{R}' & & \mathbf{e}_2 \end{array} \right),$$

where $\mathbf{R}'$ is given by (3.3) and $\mathbf{e}_2$ is a $(k_1 + k_2 - 1)$ -element column vector of all ones. In $\mathbf{R}'_1$, for each $q = k_1 + k_2, \dots, 2k_1 + 2k_2 - 2$, replace $\text{row}(q)$ with $\text{row}(q) - \text{row}(q - k_1 - k_2 + 1)$. Ignoring duplicate rows, $\mathbf{R}'_1$ transforms into

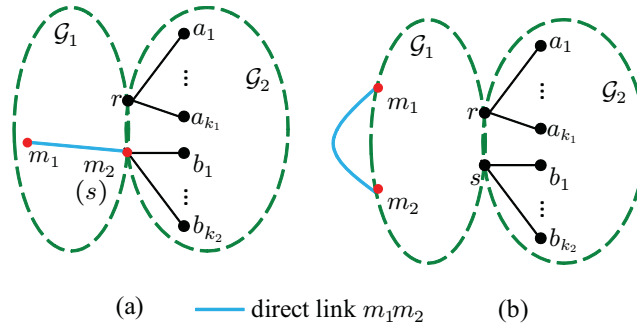
$$\mathbf{R}''_1 = \left(\begin{array}{c|c|c} \text{Exterior} & W_{l_1} & W_{l_2} \\ \text{Links} & & \\ \hline \mathbf{R}' & \mathbf{e}_2 & \\ \hline & -1 & 1 \end{array} \right),$$

where all rows are linearly independent. Since any sub-set of k ($k \geq 1$) equations in $\mathbf{R}''_1$ contains more than k unknown variables, none of these variables can be identified. When $\mathcal{G}_1$ and $\mathcal{G}_2$ are not connected, the rank of $\mathbf{R}''_1$ can be even smaller. Thus, W_{l_1} and W_{l_2} are unidentifiable, contradicting the assumption that all the interior links are identifiable.

Based on 1) and 2), we see that $\mathcal{G} - l_1$ must be 2-edge-connected for any $l_1 \in L(\mathcal{H})$ (i.e., Condition ① holds).

3) Suppose that all the interior links of $\mathcal{G}$ are identifiable, and $\mathcal{G} + m_1 m_2$ is not 3-vertex-connected. Then the *connectivity*⁴ of $\mathcal{G} + m_1 m_2$ must be 1 or 2. Before presenting the proof, one definition is required below.

⁴The greatest integer k such that $\mathcal{G}$ is k -vertex-connected is the *connectivity* of $\mathcal{G}$.

Figure 3.4: Possible scenarios of a 2-vertex cut $\{r, s\}$.**Definition 3.**

- a) A *cut-vertex* is a vertex whose removal will disconnect the graph.
- b) A *2-vertex cut* is a set of two vertices $\{v_1, v_2\}$ such that removing v_1 or v_2 alone does not disconnect $\mathcal{G}$, but removing both disconnects $\mathcal{G}$.

(3.i). Suppose that the connectivity of $\mathcal{G} + m_1m_2$ is 1. Then it must have a cut-vertex, denoted by r . There are three possible cases as illustrated in Fig. 3.3. First, if $r \in \{m_1, m_2\}$, as shown in Fig. 3.3 (a), then removing r does not disconnect $\mathcal{G} + m_1m_2$ since $\mathcal{H}$ has been assumed to be connected. Thus, r must be in $\mathcal{H}$. Let $\mathcal{G}_1 - r$ and $\mathcal{G}_2 - r$ denote two of the connected components separated by r . If each of $\mathcal{G}_1$ and $\mathcal{G}_2$ contains a monitor, as shown in Fig. 3.3 (b), then removing r does not disconnect $\mathcal{G} + m_1m_2$ because link m_1m_2 still connects $\mathcal{G}_1$ and $\mathcal{G}_2$; if one of the components, say $\mathcal{G}_2 - r$, has no monitor, as shown in Fig. 3.3 (c), then any $m_1 \rightarrow m_2$ path employing links in $\mathcal{G}_2$ must both enter and leave $\mathcal{G}_2$ through r , forming a cycle (which is forbidden). Therefore, the connectivity of $\mathcal{G} + m_1m_2$ must be greater than 1.

(3.ii). Suppose that the connectivity of $\mathcal{G} + m_1m_2$ is 2. Thus, there must be a 2-vertex cut, denoted by $\{r, s\}$. There are three possibilities. First, if $\{r, s\} = \{m_1, m_2\}$ (Fig. 3.3 (a)), then removing $\{r, s\}$ does not disconnect $\mathcal{G} + m_1m_2$ because the remaining graph $\mathcal{H}$ is still connected. Second, if $\exists$ one monitor in $\{r, s\}$, say $s = m_2$ as shown in Fig. 3.4 (a), then any $m_1 \rightarrow m_2$ path employing links in $\mathcal{G}_2$ must enter $\mathcal{G}_2$ through r and exit through m_2 . We can effectively view r and m_2 as the

new “monitors” and their adjacent links $\{ra_i\}_{i=1}^{k_1}$ and $\{m_2b_j\}_{j=1}^{k_2}$ as the new “exterior links” for $\mathcal{G}_2$. Applying Corollary 3.2 yields that $\{ra_i\}_{i=1}^{k_1}$ and $\{m_2b_j\}_{j=1}^{k_2}$ are all unidentifiable, contradicting the assumption that all the interior links are identifiable (because $\{ra_i\}_{i=1}^{k_1}$ are interior links). Finally, if both r and s are in $\mathcal{H}$, as shown in Fig. 3.4 (b), then any $m_1 \rightarrow m_2$ path employing links in $\mathcal{G}_2$ must enter/exit $\mathcal{G}_2$ via r and s . Note that there must be a component separated by $\{r, s\}$ that has no monitor (e.g., $\mathcal{G}_2$) because otherwise link m_1m_2 will keep the components connected after removing r and s . For links in $\mathcal{G}_2$, r and s are effectively the new “monitors”, and thus by Corollary 3.2 the links $\{ra_i\}_{i=1}^{k_1}$ and $\{sb_j\}_{j=1}^{k_2}$ are unidentifiable. This also contradicts the assumption that all the interior links are identifiable.

Thus, the connectivity of $\mathcal{G} + m_1m_2$ must be greater than 2, i.e., $\mathcal{G} + m_1m_2$ is 3-vertex-connected. $\square$

3.2.2 Proof of Theorem 3.3: Sufficient Part

Given Conditions ①–②, we need to show that all links in $\mathcal{H}$ are identifiable. We first introduce two types of identifiable links. The argument then consists of showing that every interior link belongs to one of these two types.

3.2.2.1 Cross-link, Shortcut, and Non-separating Cycle

We begin by introducing a few notions essential to our proof.

The first notion is a cross-link that connects nodes on two simple paths between the monitors.

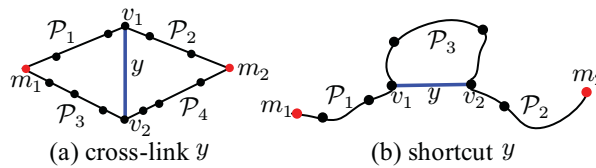


Figure 3.5: Two types of identifiable links in $\mathcal{H}$.

Definition 4. As illustrated in Fig. 3.5 (a), link y is a *cross-link* if $\exists$ four $m_1 \rightarrow m_2$ paths $\mathcal{P}_A, \mathcal{P}_B, \mathcal{P}_C$, and $\mathcal{P}_D$ formed from simple paths $\mathcal{P}_1, \dots, \mathcal{P}_4$ by:

$$\begin{cases} \mathcal{P}_A = \mathcal{P}_1 \cup \mathcal{P}_2 \\ \mathcal{P}_B = \mathcal{P}_3 \cup \mathcal{P}_4 \end{cases}, \quad \begin{cases} \mathcal{P}_C = \mathcal{P}_1 \cup y \cup \mathcal{P}_4 \\ \mathcal{P}_D = \mathcal{P}_3 \cup y \cup \mathcal{P}_2 \end{cases}, \quad (3.4)$$

such that

$$\begin{cases} |\mathcal{P}_1 \cap \mathcal{P}_2| = 1 \\ |\mathcal{P}_3 \cap \mathcal{P}_4| = 1 \end{cases}, \quad \begin{cases} |\mathcal{P}_2 \cap \mathcal{P}_3| = 0 \\ |\mathcal{P}_1 \cap \mathcal{P}_4| = 0 \end{cases}. \quad (3.5)$$

See Mathematical Notations for definitions of graph union/intersection and $|\cdot|$; note that paths are also graphs. The constraints in (3.5) are used to ensure that $\mathcal{P}_A$ – $\mathcal{P}_D$ are *simple* paths, e.g., $|\mathcal{P}_1 \cap \mathcal{P}_2| = 1$ ($\mathcal{P}_1$ and $\mathcal{P}_2$ have no common node other than v_1) ensures that no cycles exist in $\mathcal{P}_A$. However, this does not require $\mathcal{P}_1$ – $\mathcal{P}_4$ to be node disjoint, e.g., $\mathcal{P}_1$ and $\mathcal{P}_3$ can have common nodes. A cross-link y can then be identified by

$$W_y = \frac{1}{2}(W_{\mathcal{P}_C} + W_{\mathcal{P}_D} - W_{\mathcal{P}_A} - W_{\mathcal{P}_B}). \quad (3.6)$$

The second notion is a shortcut that connects the end-points of a simple path whose metric is known.

Definition 5. As illustrated in Fig. 3.5 (b), link y is a *shortcut* if $\exists$ a simple path $\mathcal{P}_3$ whose metric has been identified such that the following $m_1 \rightarrow m_2$ simple paths can be formed:

$$\mathcal{P}_A = \mathcal{P}_1 \cup y \cup \mathcal{P}_2, \quad \mathcal{P}_B = \mathcal{P}_1 \cup \mathcal{P}_3 \cup \mathcal{P}_2, \quad (3.7)$$

satisfying $|\mathcal{P}_1 \cap \mathcal{P}_3| = 1$, $|\mathcal{P}_2 \cap \mathcal{P}_3| = 1$, and $|\mathcal{P}_1 \cap \mathcal{P}_2| = 0$.

Again, the constraints are used to guarantee that $\mathcal{P}_A$ and $\mathcal{P}_B$ are simple paths. A shortcut y can be identified by

$$W_y = W_{\mathcal{P}_A} - W_{\mathcal{P}_B} + W_{\mathcal{P}_3}. \quad (3.8)$$

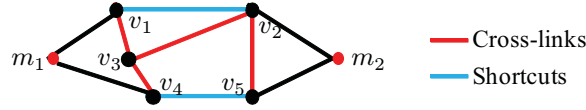
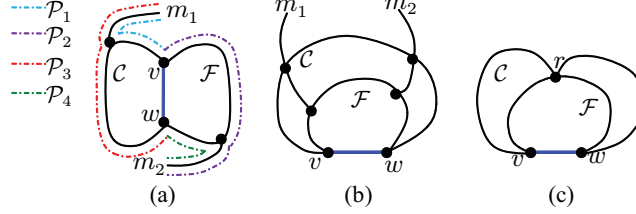


Figure 3.6: Sample network with identifiable interior graph.

Figure 3.7: Possible cases of interior link vw : (a) Case A, (b) Case B-1, (c) Case B-2 (Case B-2 is independent of the two monitor locations).

The third notion is a special kind of cycle defined as follows.

Definition 6. A *non-separating cycle* in $\mathcal{G}$, denoted by $\mathcal{F}$, is an induced sub-graph⁵ such that: (i) $\mathcal{F}$ is a cycle (see definition in Mathematical Notations), and (ii) $\mathcal{F}$ does not separate any node from monitors, i.e., each connected component in $\mathcal{G} \setminus \mathcal{F}$ contains at least one monitor.

For example, there are four non-separating cycles in Fig. 3.6: $v_1v_2v_3v_1$, $v_4v_3v_2v_5v_4$, $m_1v_1v_3v_4m_1$, and $v_5v_2m_2v_5$. Cycle $v_4v_3v_1v_2v_5v_4$ is not a non-separating cycle as it is not induced (due to link v_2v_3), neither is $v_4m_1v_1v_2v_5v_4$ as it separates v_3 from monitors.

3.2.2.2 The Proof

The key to the sufficiency proof is to show that each interior link can be categorized as either a cross-link or a shortcut when the network satisfies Conditions ① and ②. The proof proceeds in three steps.

The *first step* is to show that under Conditions ① and ②, every interior link vw satisfies one of the three cases as shown in Fig. 3.7. Specifically, we have the following lemma.

⁵An *induced sub-graph* $\mathcal{G}'$ of $\mathcal{G}$ is a sub-graph such that for any pair of vertices v and w in $\mathcal{G}'$, vw is an edge in $\mathcal{G}'$ if and only if vw is an edge in $\mathcal{G}$.

Lemma 3.4. If graph $\mathcal{G}$ satisfies Conditions ① and ②, then for any interior link vw , there exists a non-separating cycle $\mathcal{F}$ with $vw \in L(\mathcal{F})$, a cycle $\mathcal{C}$ with $vw \in L(\mathcal{C})$, a simple path $\mathcal{P}_1$ connecting one monitor with a node on $\mathcal{F} - v - w$, and a simple path $\mathcal{P}_2$ connecting the other monitor with a node on $\mathcal{C} - v - w$ such that

- (a) $\mathcal{F}$ and $\mathcal{C}$ have at most one common node other than v, w (i.e., $|V(\mathcal{F}) \cap V(\mathcal{C})| \leq 3$);
- (b) $\mathcal{P}_1$ and $\mathcal{P}_2$ are disjoint, neither going through v nor w (i.e., $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$, $v, w \notin V(\mathcal{P}_1)$ and $v, w \notin V(\mathcal{P}_2)$);
- (c) $|V(\mathcal{P}_1) \cap V(\mathcal{F})| = 1$, $|V(\mathcal{P}_2) \cap V(\mathcal{C})| = 1$.

Proof. See Section E.1 in Appendix E. □

Lemma 3.4 states that there must exist two cycles sharing link vw , a non-separating cycle $\mathcal{F}$ and a (not necessarily non-separating) cycle $\mathcal{C}$ that satisfy conditions (a)–(c) in the lemma. These conditions imply three possible cases:

- *Case A (Fig. 3.7 (a)):* $\mathcal{C}$ is also a non-separating cycle, $\mathcal{F}$ and $\mathcal{C}$ have no common node other than v and w , and each of $\mathcal{F}$ and $\mathcal{C}$ connects to a different monitor by a disjoint simple path;
- *Case B-1 (Fig. 3.7 (b)):* any path from any node in $\mathcal{F} - v - w$ to monitors must have a common node with $\mathcal{C} - v - w$;
- *Case B-2 (Fig. 3.7 (c)):* $\mathcal{F}$ and $\mathcal{C}$ must have at least one common node (denoted by r) other than v and w .

These cases are guaranteed to be complete by Lemma 3.4. We will show later that Case A links are cross-links and Case B (including B-1 and B-2) links are shortcuts.

The *second step* is to recognize that in Case A (Fig. 3.7 (a)), we can construct four measurement paths $\mathcal{P}_A$ – $\mathcal{P}_D$ by (3.4), using $\mathcal{P}_1$ – $\mathcal{P}_4$ marked in the figure. Thus, a Case A link vw is a cross-link and can be identified as in (3.6). The challenge is that it is impossible to construct measurement paths as in (3.4) for the other two

cases (Fig. 3.7 (b) and 3.7 (c)), since the conditions in (3.5) cannot be satisfied. This motivates us to explore the possibility of identifying W_{vw} in these two cases by (3.8), i.e., to investigate whether vw in Fig. 3.7 (b) and 3.7 (c) can be characterized as a shortcut.

The *third step* is to show that any link of Case B-1 or B-2 is indeed a shortcut. We prove this by showing that for any Case B link, we can find a detour path connecting its end-points such that all the links in this detour path are cross-links, and thus can be identified by (3.6). To this end, we show the following lemma.

Lemma 3.5. Let $m_1^*, m_2^* \in \{m_1, m_2\}$ with $m_1^* \neq m_2^*$ (i.e., $m_1^* = m_1, m_2^* = m_2$ or $m_1^* = m_2, m_2^* = m_1$). If $\mathcal{G}$ satisfies Conditions ① and ②, then

- (a) for any non-separating cycle in $\mathcal{G}$, there is at most *one* Case B link in this non-separating cycle;
- (b) for any Case B link vw in the interior graph of $\mathcal{G}$, there exists a non-separating cycle $\mathcal{F}_{vw}$ with $vw \in L(\mathcal{F}_{vw})$ and $m_1, m_2 \notin V(\mathcal{F}_{vw})$. For this non-separating cycle $\mathcal{F}_{vw}$, there exist disjoint simple paths $\mathcal{P}(m_1^*, v)$ and $\mathcal{P}(m_2^*, w)$, each intersecting with $\mathcal{F}_{vw}$ only at the end-point, i.e., $V(\mathcal{P}(m_1^*, v) \cap \mathcal{F}_{vw}) = \{v\}$ and $V(\mathcal{P}(m_2^*, w) \cap \mathcal{F}_{vw}) = \{w\}$.

Proof. See Section E.2 Appendix E. □

Lemma 3.5 (b) implies that a Case B link y must reside on a non-separating cycle $\mathcal{F}$ within the interior graph. Meanwhile, Lemma 3.5 (a) implies that all the other links on cycle $\mathcal{F}$ are cross-links, as there is at most one Case B link on each non-separating cycle. Furthermore, we show in Lemma 3.5 (b) that there exist disjoint simple paths $\mathcal{P}_1$ and $\mathcal{P}_2$ connecting the end-points of link y to different monitors, each sharing only one common node with $\mathcal{F}$ at the end-points of link y . Together, these allow us to construct two measurement paths $\mathcal{P}_A$ and $\mathcal{P}_B$ as in (3.7), where $\mathcal{P}_3 = \mathcal{F} - y$ consists of only cross-links whose metrics are known from (3.6). Thus,

a Case B link y is a shortcut and can be identified by (3.8).

Therefore, under Conditions ① and ②, every interior link is identifiable. $\square$

Discussions on Cross-links and Shortcuts: Fig. 3.6 displays a network satisfying Conditions ① and ②, where each interior link can be characterized as either a cross-link or a shortcut. The significance of the classification of cross-links and shortcuts is that without testing the conditions in Theorem 3.3, the identifiability of a specific link in a given network can be easily determined by testing if it is a cross-link or a shortcut. In particular, suppose we are only interested in the identifiability of link l . If l can be quickly visualized to be included in sub-graphs illustrated in Fig. 3.5 (a) or Fig. 3.5 (b), then we can deduce that link l is identifiable, and thus there is no need to test the conditions in Theorem 3.3.

Since we are only able to identify the interior graph in a given network with two monitors, the next question we seek to answer in the following section is if it is possible to identify the entire network using more monitors.

3.3 Identifiability with Three or More Monitors

Although two monitors are not sufficient to identify all link metrics in $\mathcal{G}$, we explore in this section the case of identifying the entire network using three or more monitors.

3.3.1 Conversion into Two-Monitor Problem

Section 3.2 suggests that it is easier to identify links that are one-hop away from the monitors. This observation motivates us to construct an extended graph $\mathcal{G}_{ex}$ of $\mathcal{G}$. As illustrated in Fig. 3.8, given a graph $\mathcal{G}$ with κ monitors, its extended graph $\mathcal{G}_{ex}$ is obtained by adding two *virtual monitors* m_1^+ and m_2^+ , and 2κ *virtual links* between each pair of virtual-actual monitors. In this way, all links of interest (actual links in $\mathcal{G}$) are at least one-hop away from virtual monitors m_1^+ and m_2^+ . This construction

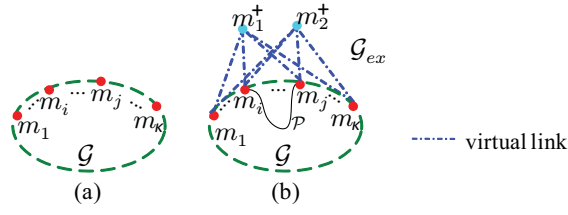


Figure 3.8: (a) $\mathcal{G}$ with κ ($\kappa \geq 3$) monitors; (b) $\mathcal{G}_{ex}$ with two virtual monitors: m_1^+ and m_2^+ .

immediately converts the problem of identifying $\mathcal{G}$ using κ monitors to a problem of identifying the interior graph of $\mathcal{G}_{ex}$ using two monitors (again we have no prior knowledge of link metrics in $\mathcal{G}$ or $\mathcal{G}_{ex}$). Therefore, we can apply Theorem 3.3 to obtain the following result.

Lemma 3.6. Employing κ ($\kappa \geq 3$) monitors to measure simple paths, the necessary and sufficient condition on the network topology $\mathcal{G}$ for identifying all link metrics in $\mathcal{G}$ is that the associated extended graph $\mathcal{G}_{ex}$ has an identifiable interior graph, i.e., $\mathcal{G}_{ex}$ satisfies Conditions ① and ② in Theorem 3.3.

Proof. Since $\mathcal{G}$ is the interior graph of $\mathcal{G}_{ex}$, it suffices to show that the information attainable by the real monitors $m_1, \dots, m_\kappa$ is the same as the information attainable by the virtual monitors m_1^+ and m_2^+ , if the virtual monitors can make end-to-end measurements along simple paths in $\mathcal{G}_{ex}$.

First, we show that any measurement between the real monitors can be obtained from measurements between m_1^+ and m_2^+ . To this end, consider a path $m_i \mathcal{P} m_j$ ($i, j \in \{1, \dots, \kappa\}, i \neq j$) in $\mathcal{G}$, as shown in Fig. 3.8 (b). Four simple paths between m_1^+ and m_2^+ can be constructed:

$$\left\{ \begin{array}{l} \mathcal{P}_A = m_1^+ m_i m_2^+, \\ \mathcal{P}_B = m_1^+ m_j m_2^+, \\ \mathcal{P}_C = m_1^+ m_i \mathcal{P} m_j m_2^+, \\ \mathcal{P}_D = m_1^+ m_j \mathcal{P} m_i m_2^+. \end{array} \right. \quad (3.9)$$

Viewing $m_i \mathcal{P} m_j$ as a “cross-link”, we can compute $W_{m_i \mathcal{P} m_j}$ from the measurements on these four paths via (3.6) (replacing W_y by $W_{m_i \mathcal{P} m_j}$).

Moreover, we show that measurements between m_1^+ and m_2^+ in $\mathcal{G}_{ex}$ do not provide extra information for identifying links in $\mathcal{G}$ compared with measurements attainable by the real monitors. This is proved by observing that for any $m_1^+ \rightarrow m_2^+$ simple path $m_1^+ m_i \mathcal{P} m_j m_2^+$ ($i, j \in \{1, \dots, \kappa\}, i \neq j$) containing at least one link in $\mathcal{G}$, the information relevant for identifying links in $\mathcal{G}$ can be obtained by measuring its $m_i \rightarrow m_j$ sub-path $m_i \mathcal{P} m_j$, which must also be a simple path. $\square$

3.3.2 Complete Identifiability Condition

The special structure of $\mathcal{G}_{ex}$ allows us to consolidate the two Conditions ① and ② into a single condition, stated as follows.

Theorem 3.7. Assume that κ ($\kappa \geq 3$) monitors are used to measure simple paths. The necessary and sufficient condition on the network topology $\mathcal{G}$ for identifying all link metrics in $\mathcal{G}$ is that the associated extended graph $\mathcal{G}_{ex}$ be 3-vertex-connected.

Proof. To prove Theorem 3.7, we first have two claims.

1) **Claim A.** Given a graph $\mathcal{G}$ employing κ ($\kappa \geq 3$) monitors, the extended graph $\mathcal{G}_{ex}$ of $\mathcal{G}$ satisfies Conditions ① (i.e., $\mathcal{G}_{ex} - l$ is 2-edge-connected for each link l in $\mathcal{G}$) if and only if $\mathcal{G}_{ex}$ is 3-edge-connected.

Claim A is proved as follows:

Necessary part.

Suppose $\mathcal{G}_{ex} - l$ is 2-edge-connected for all l in $\mathcal{G}$. Consider removing two links in $\mathcal{G}_{ex}$, denoted by l_1 and l_2 .

(a) If at least one of these links, say l_1 , is in $L(\mathcal{G})$, then by assumption $\mathcal{G}_{ex} - l_1$ is 2-edge-connected. Thus, $\mathcal{G}_{ex} - l_1 - l_2$ is connected.

(b) Suppose none of these links is in $\mathcal{G}$, i.e., both l_1 and l_2 are virtual links.

Since the virtual monitors m_1^+ and m_2^+ each connect to all actual monitors in $\mathcal{G}$, and there are at least 3 actual monitors, m_1^+ and m_2^+ are each connected to $\mathcal{G}$ via at least 3 virtual links. Therefore, m_1^+ and m_2^+ are still connected with $\mathcal{G}$ after l_1 and l_2 are deleted. Since we have assumed $\mathcal{G}$ to be a connected graph, $\mathcal{G}_{ex} - l_1 - l_2$ is connected.

We have shown that $\mathcal{G}_{ex}$ remains connected after removing any two links. Therefore, $\mathcal{G}_{ex}$ is 3-edge-connected when $\mathcal{G}_{ex} - l$ ($l \in L(\mathcal{G})$) is 2-edge-connected.

Sufficient part.

Suppose $\mathcal{G}_{ex}$ is 3-edge-connected. Then obviously, $\mathcal{G}_{ex} - l$ is 2-edge-connected for each $l \in L(\mathcal{G})$.

2) **Claim B.** Given a graph G employing κ ($\kappa \geq 3$) monitors, the extended graph $\mathcal{G}_{ex}$ of $\mathcal{G}$ satisfies Conditions ② (i.e., $\mathcal{G}_{ex} + m_1^+ m_2^+$ is 3-vertex-connected) if and only if $\mathcal{G}_{ex}$ is 3-vertex-connected.

Claim B is proved as follows:

Necessary part.

We prove the necessary part by contradiction. Suppose $\mathcal{G}_{ex}$ is not 3-vertex-connected, but $\mathcal{G}_{ex} + m_1^+ m_2^+$ is 3-vertex-connected, then the connectivity of $\mathcal{G}_{ex}$ must be 2, because removing one link will decrease connectivity by at most 1. Thus, there must exist two nodes, denoted by v_1 and v_2 , whose removal will disconnect $\mathcal{G}_{ex}$. There are 3 possibilities for v_1 and v_2 .

(a) If v_1, v_2 are m_1^+, m_2^+ , then after their removal, the remaining graph ($\mathcal{G}$) is still connected.

(b) If v_1 is a virtual monitor (m_1^+ or m_2^+) and v_2 is a node in $\mathcal{G}$, then $\mathcal{G}_{ex} - v_1 - v_2$ being disconnected will imply $\mathcal{G}_{ex} + m_1^+ m_2^+ - v_1 - v_2$ also being disconnected (as the remaining graphs of $\mathcal{G}_{ex}$ and $\mathcal{G}_{ex} + m_1^+ m_2^+$ are the same), contradicting the assumption that $\mathcal{G}_{ex} + m_1^+ m_2^+$ is 3-vertex-connected.

(c) If v_1, v_2 are both in $\mathcal{G}$ (can be real monitors), then two cases may occur after removing v_1 and v_2 : (a) $\exists$ a connected component that does not contain any

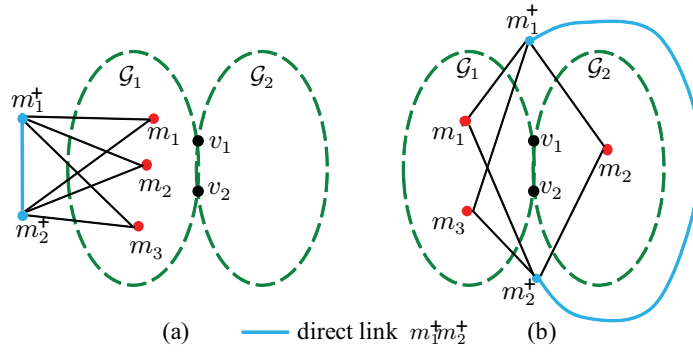


Figure 3.9: $\mathcal{G}_{ex} - v_1 - v_2$ is disconnected, where $v_1, v_2 \in V(\mathcal{G})$.

real monitor; (b) each connected component contains at least one real monitor, as illustrated in Fig. 3.9. In the case of Fig. 3.9 (a), $\mathcal{G}_{ex} + m_1^+m_2^+ - v_1 - v_2$ is disconnected as well, contradicting the 3-vertex-connectivity of $\mathcal{G}_{ex} + m_1^+m_2^+$. In the case of Fig. 3.9 (b), different components in $\mathcal{G}_{ex} - v_1 - v_2$ can still connect via virtual links and virtual monitors, thus contradicting the assumption that $\mathcal{G}_{ex} - v_1 - v_2$ is disconnected.

Hence, when $\mathcal{G}_{ex} + m_1^+m_2^+$ is 3-vertex-connected, the connectivity of $\mathcal{G}_{ex}$ cannot be less than 3, i.e., $\mathcal{G}_{ex}$ is also 3-vertex-connected.

Sufficient part.

If $\mathcal{G}_{ex}$ is 3-vertex-connected, then after adding one link $m_1^+m_2^+$, $\mathcal{G}_{ex} + m_1^+m_2^+$ is also 3-vertex-connected.

3) From the structure of $\mathcal{G}_{ex}$ (see Fig. 3.8), Claims A and B suggest that $\mathcal{G}_{ex}$ satisfies Conditions ① and ② in Theorem 3.3 if and only if $\mathcal{G}_{ex}$ is both 3-edge-connected and 3-vertex-connected. According to Proposition 1.4.2 in [47], a 3-vertex-connected graph is also 3-edge-connected. Thus, the necessary and sufficient conditions in Lemma 3.6 can be simplified to a single condition that $\mathcal{G}_{ex}$ be 3-vertex-connected. $\square$

3.4 Testing and Ensuring Identifiability

The conditions we have derived have broader impact than mere theoretical interest. A major benefit of characterizing network identifiability in terms of network topology is that we can leverage existing graph-processing algorithms to efficiently test for the identifiability of a given network. In this section, we present efficient algorithms that can test for the identifiability of a given network with a given monitor placement.

3.4.1 Efficient Test for Interior Graph Identifiability

Given a network topology $\mathcal{G}$ with two monitors, we know from Theorem 3.1 that it is impossible to identify the entire $\mathcal{G}$. Nevertheless, we can test whether the interior graph is identifiable using Conditions ① and ② in Theorem 3.3, which transform into multiple tests of edge/vertex connectivity. The problem of determining whether a given graph is k -edge/vertex-connected has been well studied. Specifically, fast algorithms have been proposed to test if a graph is: (i) 2-edge-connected [50], or (ii) 3-vertex-connected [51], both in time $O(|V| + |L|)$ ($|V|$: number of nodes; $|L|$: number of links). In [50], a spanning tree is used for testing the existence of bridges in the network; if no bridges exist, then the network is 2-edge-connected. While for testing a network $\mathcal{G}$ be 3-vertex-connected, [51] first decomposes $\mathcal{G}$ into triconnected components (see Definition 8); then the network is 3-vertex-connected if and only if the number of triconnected components is only one. Using these algorithms, we can test for the identifiability of the *interior graph* of $\mathcal{G}$ as follows:

- (1) For each interior link l , apply the 2-edge-connectivity test in [50] to $\mathcal{G} - l$. $\mathcal{G}$ is unidentifiable if the test fails;
- (2) Apply the 3-vertex-connectivity test in [51] to $\mathcal{G} + m_1m_2$. $\mathcal{G}$ is unidentifiable if the test fails.

The interior graph of $\mathcal{G}$ is identifiable if all the tests succeed. The overall complexity is $O(|L|(|V| + |L|))$.

3.4.2 Efficient Test for Complete Identifiability

Similarly, for network $\mathcal{G}$ with three or more monitors, we can test for the identifiability of the *entire* $\mathcal{G}$ using the condition in Theorem 3.7:

- (1) Construct the extended graph $\mathcal{G}_{ex}$ as in Fig. 3.8;
- (2) Apply the 3-vertex-connectivity test in [51] to $\mathcal{G}_{ex}$. $\mathcal{G}$ is identifiable if the test succeeds, and unidentifiable otherwise.

The complexity of this algorithm is $O(|V(\mathcal{G}_{ex})| + |L(\mathcal{G}_{ex})|)$, which is the same as⁶ $O(|V(\mathcal{G})| + |L(\mathcal{G})|)$.

In this section, necessary and sufficient topological conditions are established to characterize the identifiability of the interior and the entire graph under the assumption that the network graph is undirected and all measurement paths are arbitrarily controllable without containing cycles. Besides the theoretical value, these conditions are also efficiently verifiable using existing algorithms in graph theory. Moreover, in networks with two monitors being employed, the concepts of cross-link and shortcut used for the theorem proof can also assist in fast determining the identifiability of the given set of links without verifying the satisfaction of Conditions ① and ② in Theorem 3.3. We note that in cases where more routing constraints are imposed to the network (e.g., controllable routing is not fully supported), the conditions developed in this chapter are only necessary but not sufficient. Nevertheless, as necessary conditions for networks under constrained routing mechanisms, they are still capable of revealing some insights regarding network internal states. Specifically, when these necessary conditions are not satisfied, we can assert that the network

⁶This is because $|V(\mathcal{G}_{ex})| + |L(\mathcal{G}_{ex})| = |V(\mathcal{G})| + 2 + |L(\mathcal{G})| + 2\kappa \leq |V(\mathcal{G})| + 2 + |L(\mathcal{G})| + 2|V(\mathcal{G})| = 3|V(\mathcal{G})| + |L(\mathcal{G})| + 2$.

(or the interior graph) is not completely identifiable.

Path Construction and Link Identification for Completely Identifiable Networks

After verifying that the network is identifiable, i.e., there exist n (n is the number of links in a network) linearly independent simple paths between monitors, the natural followup questions are: how can we efficiently find n such paths, and how can we efficiently compute link metrics from measurements on these paths? In this chapter, we first provide the key idea behind our solutions and then formally present the algorithms and analyze their complexities.

4.1 Algorithm Overview

The idea of efficient path construction and link identification originates from the identifiability condition in Theorem 3.7. Since the condition is necessary, any identifiable network $\mathcal{G}$ must have a 3-vertex-connected extended graph $\mathcal{G}_{ex}$. We further

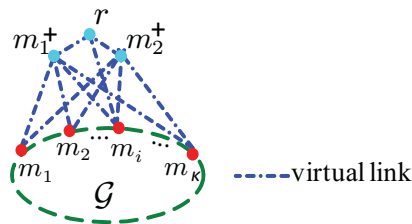


Figure 4.1: The r -extended graph $\mathcal{G}_{ex}^*$ of $\mathcal{G}$.

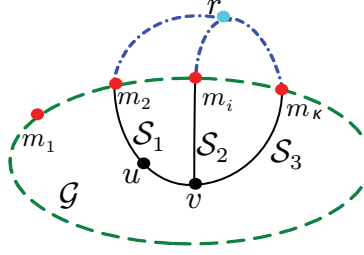


Figure 4.2: Three internally vertex disjoint paths between v and r imply three monitor-to-monitor simple paths: $\mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{S}_1 \cup \mathcal{S}_3$, and $\mathcal{S}_2 \cup \mathcal{S}_3$.

extend $\mathcal{G}_{ex}$ by adding another virtual node r and connecting it to the virtual monitors m_1^+ , m_2^+ and any one of the real monitors m_i (for any $i \in \{1, \dots, \kappa\}$) with three virtual links, as illustrated in Fig. 4.1; we refer to the new graph as the r -extended graph, denoted by $\mathcal{G}_{ex}^*$. It can be shown that $\mathcal{G}_{ex}^*$ is also 3-vertex-connected. Menger's theorem [47] shows that there exist at least k internally vertex disjoint simple paths (i.e., the paths are disjoint except at end-points) between any two nodes in a k -vertex-connected graph. Therefore, it implies that there exist 3 internally vertex disjoint simple paths between any two nodes in $\mathcal{G}_{ex}^*$. In particular, as illustrated in Fig. 4.2, each non-monitor node v has (at least) 3 internally vertex disjoint simple paths to the virtual node r . Since any path to r must go through at least one (real) monitor, we can truncate each path at the first monitor on the way to r (i.e., removing the sub-path from this monitor to r), which provides three v -to-monitor paths $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ that are disjoint except at v . This allows us to construct three monitor-to-monitor paths $\mathcal{P}_1 := \mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{P}_2 := \mathcal{S}_1 \cup \mathcal{S}_3$, and $\mathcal{P}_3 := \mathcal{S}_2 \cup \mathcal{S}_3$, each being a simple path valid for taking measurements. Based on measurements $W_{\mathcal{P}_i}$ ($i = 1, 2, 3$) from the constructed paths, we can obtain the individual metrics of $\mathcal{S}_i$ by solving the following linear equations:

$$\begin{cases} W_{\mathcal{S}_1} + W_{\mathcal{S}_2} = W_{\mathcal{P}_1}, \\ W_{\mathcal{S}_1} + W_{\mathcal{S}_3} = W_{\mathcal{P}_2}, \\ W_{\mathcal{S}_2} + W_{\mathcal{S}_3} = W_{\mathcal{P}_3}, \end{cases}$$

where W_{S_i} ($i = 1, 2, 3$) is the path metric on S_i . Repeating this procedure for every node in $\mathcal{G}$ yields the metrics from each node to three monitors. Furthermore, we will show a way to construct these paths such that they are nested, i.e., if a path from v to monitor m_2 goes through a neighbor u , as illustrated in Fig. 4.2, then u must use the same path to connect to m_2 . Therefore, we can calculate the metric of link uv by subtracting the $u \rightarrow m_2$ path metric from the $v \rightarrow m_2$ path metric. We now present the algorithms in detail.

4.2 Spanning Tree-based Path Construction

Given an arbitrary network $\mathcal{G}$, we propose an algorithm, *Spanning Tree-based Path Construction (STPC)*, to construct linearly independent monitor-to-monitor paths, so that links can be uniquely identified from measurements on these paths. We assume that $\mathcal{G}$ is identifiable, which can be verified by Theorem 3.7. In essence, STPC exploits a property of identifiable networks in terms of spanning trees. To this end, we introduce the following definition.

Definition 7. Two spanning trees of an undirected graph $\mathcal{G}(V, L)$ are *independent with respect to* (w.r.t.) a vertex $r \in V$ if the paths from v to r along these trees are internally vertex disjoint for every vertex $v \in V$ ($v \neq r$).

For a 3-vertex-connected graph and any given vertex w , Theorem 6 in [52] states that there exist three spanning trees that are pairwise independent w.r.t. w . In particular, since the r -extended graph $\mathcal{G}_{ex}^*$ is guaranteed to be 3-vertex-connected for any identifiable network, we will always be able to find three spanning trees in $\mathcal{G}_{ex}^*$ that are pairwise independent w.r.t. r . These spanning trees provide three internally vertex disjoint paths from each non-monitor node v to r . As previously illustrated in Fig. 4.2, STPC constructs measurement paths by truncating these node-to- r paths at monitors (to get S_1 , S_2 , and S_3 illustrated in Fig. 4.2) and then concatenating each

Algorithm 4.1: Spanning Tree-based Path Construction (STPC)

input : Network $\mathcal{G}$ with κ monitors such that every link in $\mathcal{G}$ is identifiable
output: Measurement paths in the form of (rows in) a measurement matrix $\mathbf{R}$

- 1 $\mathbf{R} = \emptyset$;
- 2 Construct $\mathcal{G}_{ex}^*$ from $\mathcal{G}$; //see Fig. 4.1
- 3 Find three spanning trees $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ of $\mathcal{G}_{ex}^*$ that are pairwise independent w.r.t. r by the algorithm in [52];
- 4 **foreach** node v in $\mathcal{G}$ **do**
- 5 **if** v is a monitor **then**
- 6 $\mathcal{P}_{v1} \leftarrow \mathcal{S}_{v1}; \mathcal{P}_{v2} \leftarrow \mathcal{S}_{v2}; \mathcal{P}_{v3} \leftarrow \mathcal{S}_{v3}$;
- 7 **else**
- 8 $\mathcal{P}_{v1} \leftarrow \mathcal{S}_{v1} \cup \mathcal{S}_{v2}; \mathcal{P}_{v2} \leftarrow \mathcal{S}_{v2} \cup \mathcal{S}_{v3}; \mathcal{P}_{v3} \leftarrow \mathcal{S}_{v3} \cup \mathcal{S}_{v1}$;
- 9 **end**
- 10 Append all non-degenerate $\mathcal{P}_{vi}$ ($i = 1, 2, 3$) to $\mathbf{R}$;
- 11 **end**
- 12 **foreach** link l not in $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ **do**
- 13 Find a simple monitor-to-monitor path $\mathcal{P}_l$ traversing l in graph $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 + l$ (see Algorithm 4.2);
- 14 Append $\mathcal{P}_l$ to $\mathbf{R}$;
- 15 **end**

pair of truncated paths to form simple monitor-to-monitor paths (i.e., paths $\mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{S}_2 \cup \mathcal{S}_3$, and $\mathcal{S}_1 \cup \mathcal{S}_3$ in Fig. 4.2) in the original graph $\mathcal{G}$. See Algorithm 4.1 for details.

Specifically, STPC has two main steps: (i) constructing measurement paths based on independent spanning trees, and (ii) constructing additional paths to measure links not in any of the trees. The first step begins with the application of the algorithm in [52] to find three spanning trees (see the algorithm in Appendix B) $\mathcal{T}_i$ ($i = 1, 2, 3$) of $\mathcal{G}_{ex}^*$ that are independent w.r.t. r (line 3). Based on these spanning trees, STPC constructs paths to measure links in the trees (lines 4-11). Let $\mathcal{S}_{vi}$ ($i = 1, 2, 3$) denote a simple path from node v to the first monitor m ($m \neq v$) toward r in $\mathcal{T}_i$. If no such m exists, then $\mathcal{S}_{vi}$ represents a degenerate path containing just a single node v . STPC iterates among all nodes in $\mathcal{G}$: if v is a monitor, then $\mathcal{S}_{vi}$ ($i = 1, 2, 3$) are already monitor-to-monitor simple paths (line 6); if v is not a monitor, then pairs of $\mathcal{S}_{vi}$'s again form monitor-to-monitor simple paths, as $\mathcal{S}_{v1}$, $\mathcal{S}_{v2}$, and $\mathcal{S}_{v3}$ are disjoint except at v (line 8). Therefore, all the constructed paths $\mathcal{P}_{vi}$ ($i = 1, 2, 3$) that are

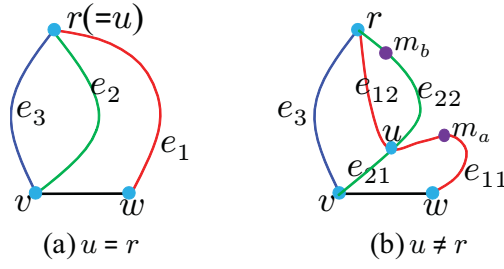


Figure 4.3: Constructing measurement path traversing a non-tree link vw .

non-degenerate (i.e., containing at least one link) are valid measurement paths, and are thus added to the measurement matrix (line 10).

The second step constructs paths for links not in any of the three trees (lines 12-15). For a non-tree link l , i.e., l not in any $\mathcal{T}_i$ ($i = 1, 2, 3$), STPC invokes an auxiliary algorithm, Algorithm 4.2, to construct a measurement path $\mathcal{P}_l$ through l such that all the other links on this path belong to the trees. Algorithm 4.2 utilizes a simple observation as follows. As illustrated in Fig. 4.3, among the three internally vertex disjoint paths from v to r along the three spanning trees, there exist at least two paths, say ve_2r and ve_3r , that do not traverse w ; similarly, there exists at least one path from w to r , say we_1r , that does not traverse v (line 1). Starting from w , we follow we_1r until the first intersection u with either ve_2r or ve_3r (line 2). If $u = r$ as in Fig. 4.3 (a), then we_1r and ve_2r (or ve_3r) are disjoint except at r . Truncating these paths at the first monitors toward r provides two disjoint paths, $\mathcal{S}_{we_1r}$ and $\mathcal{S}_{ve_2r}$, that connect w and v to monitors. Connecting these paths by link vw gives a simple path between monitors that traverses only vw and links in the trees (line 4). If $u \neq r$, we assume without loss of generality that u is an internal node on ve_2r as illustrated in Fig. 4.3 (b), which divides path we_1r into sub-paths $we_{11}u$ and $ue_{12}r$, and ve_2r into $ve_{21}u$ and $ue_{22}r$. The new path formed by $we_{11}ue_{22}r$ is disjoint with ve_3r except at r . Truncating paths $we_{11}ue_{22}r$ and ve_3r again provides two disjoint paths $\mathcal{S}_{we_{11}ue_{22}r}$ and $\mathcal{S}_{ve_3r}$ connecting w and v to monitors, which together with link vw form a simple monitor-to-monitor path traversing only vw and links in the trees (line 6). The validity of the algorithm is guaranteed by the following lemma.

Algorithm 4.2: Path Construction for Non-Tree Links

input : Trees $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ constructed in Algorithm 4.1 and a link $l = vw$ not in the trees

output: A simple monitor-to-monitor path $\mathcal{P}_l$ traversing l and links in the trees

- 1 From the trees, find two paths ve_2r and ve_3r from v to r that do not traverse w , and a path we_1r from w to r that does not traverse v ;
- 2 On path we_1r starting from w , find the first intersection node u with either ve_2r or ve_3r ;
- 3 **if** $u = r$ **then**
- 4 $\mathcal{P}_l \leftarrow \mathcal{S}_{ve_2r} \cup l \cup \mathcal{S}_{we_1r}$;
- 5 **else**
- 6 $\mathcal{P}_l \leftarrow \mathcal{S}_{ve_3r} \cup l \cup \mathcal{S}_{we_{11}ue_{22}r}$;
- 7 **end**

Lemma 4.1. Path $\mathcal{P}_l$ constructed by Algorithm 4.2 is a simple monitor-to-monitor path traversing only l and links in the trees.

Proof. It is easy to see that $\mathcal{P}_l$ contains only one non-tree link, link l . To verify it as a simple path, it suffices to show that the following paths are disjoint except at r : ve_2r and we_1r if $u = r$, or ve_3r and $we_{11}ue_{22}r$ if $u \neq r$. The former is trivially satisfied. For the latter, note that sub-path $we_{11}u$ is disjoint with ve_3r as u is the first intersection (and ve_3r cannot contain u since it is internally vertex disjoint with ve_2r). Moreover, sub-path $ue_{22}r$ is disjoint with ve_3r except at r as ve_2r and ve_3r are internally vertex disjoint. Thus, paths $we_{11}ue_{22}r$ and ve_3r are disjoint except at r , completing the proof. $\square$

The correctness of STPC, i.e., the constructed measurement matrix $\mathbf{R}$ has rank n , will be clear in Section 4.3 when we present an algorithm to explicitly compute all link metrics from measurements on these paths. In general, further processing is needed to find n linearly independent rows in $\mathbf{R}$ that specify the final set of measurement paths, since $\mathbf{R}$ may contain superfluous paths. However, we have a stronger result showing that the final processing is as simple as eliminating duplicate paths because all distinct paths (paths that differ in at least one node) found by STPC are linearly independent.

Theorem 4.2. The number of distinct paths constructed by STPC equals n , the number of links in $\mathcal{G}$.

Proof. See Section E.3 in Appendix E. □

Since only distinct paths can be linearly independent, and the number of linearly independent paths constructed by STPC equals n (as these paths can uniquely identify the n links; see Section 4.3), Theorem 4.2 implies that all distinct paths found by STPC are linearly independent. Removing duplicate rows in the constructed $\mathbf{R}$ thus generates an $n \times n$ invertible measurement matrix.

4.3 Spanning Tree-based Link Identification

Given the paths constructed by STPC, we are now ready to take measurements and compute link metrics. A straightforward approach is to invert the measurement matrix to solve for the link metrics by $\mathbf{w} = \mathbf{R}^{-1}\mathbf{c}$ (assuming $\mathbf{R}$ is invertible)¹, with a complexity of $O(n^3)$, which we seek to avoid. Moreover, computing the matrix inversion may incur rounding errors [53]. We will show that the $\mathbf{R}$ generated by STPC has a special structure that allows us to directly compute link metrics at a much lower complexity. The algorithm, *Spanning Tree-based Link Identification (STLI)*, consists of three main steps as shown in Algorithm 4.3: (i) computing node-to-monitor path metrics (lines 1-3), (ii) identifying links in the spanning trees (lines 4-12), and (iii) identifying the other links (lines 13-15). In the sequel, we use $W_{\mathcal{P}}$ to denote both measured path metric if $\mathcal{P}$ is a monitor-to-monitor path, and calculated path metric if $\mathcal{P}$ is a node-to-monitor path (although the input vector $\mathbf{c}$ only contains measured path metrics).

The first step of STLI aims at computing the node-to-monitor metric $W_{\mathcal{S}_{v_i}}$ for every node $v \in V$ and every $i \in \{1, 2, 3\}$ (lines 1-3). From the path construction in

¹Existing algorithms, e.g., Gaussian elimination [12], can directly solve for $\mathbf{w}$ without computing $\mathbf{R}^{-1}$; their complexity, however, remain $O(n^3)$.

Algorithm 4.3: Spanning Tree-based Link Identification (STLI)

input : Measurement paths and spanning trees $\mathcal{T}_i$ ($i = 1, 2, 3$) constructed by Algorithm 4.1, measurements $\mathbf{c}$ on the paths
output: Vector $\mathbf{w}$ of link metrics in $\mathcal{G}$

```

1 foreach node  $v$  in  $\mathcal{G}$  do
2   | Compute  $W_{\mathcal{S}_{vi}}$  from measurements  $W_{\mathcal{P}_{vi}}$  ( $i = 1, 2, 3$ ) by (4.1);
3 end
4 foreach tree  $\mathcal{T}_i$  ( $i = 1, 2, 3$ ) do
5   | foreach link  $vw$  in tree  $\mathcal{T}_i$  ( $v$  is closer to  $r$ ) do
6     | if  $v$  is a monitor then
7       |    $W_{vw} = W_{\mathcal{S}_{wi}}$ ;
8     | else
9       |    $W_{vw} = W_{\mathcal{S}_{wi}} - W_{\mathcal{S}_{vi}}$ ;
10    | end
11  | end
12 end
13 foreach link  $l$  not in  $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$  do
14   | Compute  $W_l$  by subtracting metrics of the other links on  $\mathcal{P}_l$  from  $W_{\mathcal{P}_l}$ ;
15 end

```

steps 6 and 8 of STPC, we see that $W_{\mathcal{S}_{vi}}$ is directly measured if v is a monitor. If v is not a monitor, we can construct three linear equations based on measurements on $\mathcal{P}_{vi}$ ($i = 1, 2, 3$):

$$\begin{cases} W_{\mathcal{S}_{v1}} + W_{\mathcal{S}_{v2}} = W_{\mathcal{P}_{v1}}, \\ W_{\mathcal{S}_{v2}} + W_{\mathcal{S}_{v3}} = W_{\mathcal{P}_{v2}}, \\ W_{\mathcal{S}_{v3}} + W_{\mathcal{S}_{v1}} = W_{\mathcal{P}_{v3}}, \end{cases} \quad (4.1)$$

from which we can compute $W_{\mathcal{S}_{vi}}$ ($i = 1, 2, 3$).

The second step aims at solving for metrics of all links in the trees (lines 4-12). Consider a link vw in tree $\mathcal{T}_i$ ($i \in \{1, 2, 3\}$), where v is one hop closer to r . If v is a monitor, then the node-to-monitor path $\mathcal{S}_{wi}$ will only contain link vw , and thus its metric is also the metric of vw (line 7). If v is not a monitor, then the node-to-monitor path $\mathcal{S}_{vi}$ must be a sub-path of $\mathcal{S}_{wi}$, shorter by just link vw , and thus the difference in their metrics equals the metric of vw (line 9).

The final step addresses links not in the trees (lines 13-15). Since the measurement path $\mathcal{P}_l$ for each non-tree link l only contains l and links in the trees (Lem-

ma 4.1), we just need to subtract from $W_{\mathcal{P}_l}$ the metrics of these tree links as identified in the second step to compute the metric of l (line 14).

Besides computing link metrics, STLI also serves as a constructive proof that the paths constructed by STPC can uniquely identify all links, i.e., the generated $\mathbf{R}$ has rank n .

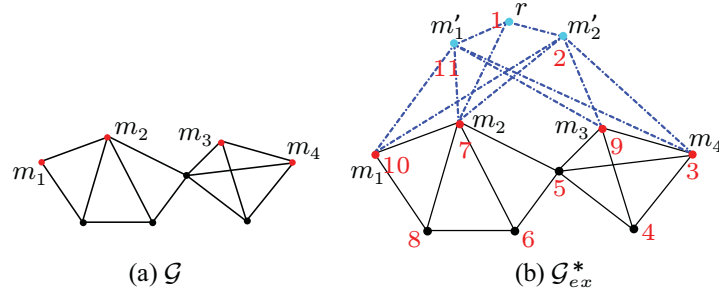
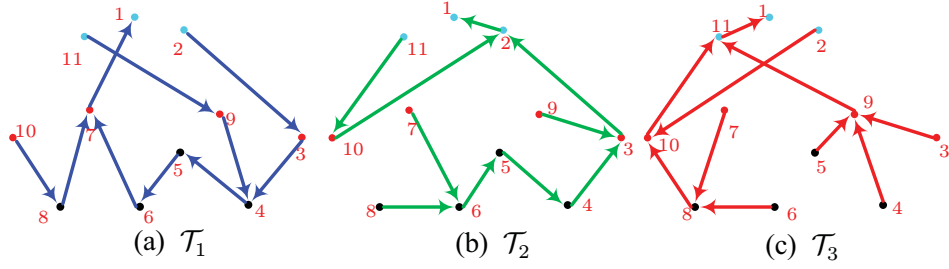
4.4 Complexity Analysis

We now analyze the complexity of the proposed algorithms. STPC has an overall complexity of $O(|V| \cdot |L|)$. Specifically, the complexity of spanning tree construction in line 3 is $O(|V| \cdot |L(\mathcal{G}_{ex}^*)|)$ [52], which is $O(|V| \cdot |L|)$ since $|L(\mathcal{G}_{ex}^*)| = O(|L(\mathcal{G})|) = O(|L|)$. For lines 4-11, paths $\mathcal{P}_{vi}$ ($i = 1, 2, 3$) can be constructed in $O(|V|)$ time for each node v ; thus, lines 4-11 take $O(|V|^2)$ time. Finally, lines 12-15 invoke Algorithm 4.2 $O(|L|)$ times, and each invocation takes time $O(|V|)$. Combining the above yields an overall complexity of $O(|V| \cdot |L|)$. We point out that it is possible to save some computation by removing redundant links in $\mathcal{G}_{ex}^*$ using an $O(|V| + |L|)$ -time algorithm² in Section 4 of [52], which reduces the number of links to $O(|V|)$ while maintaining the 3-vertex-connectivity. This step reduces the complexity of spanning tree construction to $O(|V|^2)$, but the overall complexity remains the same.

Given measurements on the constructed paths, STLI can compute link metrics in $O(|V| + |L|)$ time. Specifically, computing $W_{S_{vi}}$ (lines 1-3) takes $O(|V|)$ time. Then computing the metric of each link in the spanning trees (lines 6-10) takes only constant time, and there are $O(|V|)$ links in the trees, making the complexity of lines 4-12 $O(|V|)$. Finally, computing the metrics of non-tree links (lines 13-15) takes $O(|L|)$ time as explained below. Thus, the overall complexity is $O(|V| + |L|)$.

At first sight, it may seem that line 14 takes $O(|V|)$ time as there are $O(|V|)$ link

²This algorithm does a depth-first search and delete non-tree links as many as possible in linear time.

Figure 4.4: Sample network and its r -extended graph.Figure 4.5: Spanning trees of $\mathcal{G}_{ex}^*$ in Fig. 4.4 (b), independent w.r.t. r (node 1). Here arrows are only used to illustrate paths to r ; the original links are undirected.

metrics to subtract, making the overall complexity of STLI $O(|V| \cdot |L|)$. However, we observe that this step can be implemented in constant time using the knowledge of $W_{S_{vi}}$ as follows. Consider the two cases in constructing $\mathcal{P}_l$ as illustrated in Fig. 4.3. Suppose that path we_1r belongs to tree $\mathcal{T}_{i_1}$, ve_2r to $\mathcal{T}_{i_2}$, and ve_3r to $\mathcal{T}_{i_3}$ ($i_1, i_2, i_3 \in \{1, 2, 3\}$, $i_2 \neq i_3$). In case Fig. 4.3 (a), the measurement path $\mathcal{P}_l$ is a concatenation of link l , path $\mathcal{S}_{vi_2}$, and path $\mathcal{S}_{wi_1}$, and thus the metric of link l can be computed by $W_l = W_{\mathcal{P}_l} - W_{\mathcal{S}_{vi_2}} - W_{\mathcal{S}_{wi_1}}$. In case Fig. 4.3 (b), if the first monitor along path $we_{11}ue_{22}r$ appears before or at u (e.g., m_a), then $\mathcal{P}_l$ consists of l , $\mathcal{S}_{vi_3}$, and $\mathcal{S}_{wi_1}$, and thus $W_l = W_{\mathcal{P}_l} - W_{\mathcal{S}_{vi_3}} - W_{\mathcal{S}_{wi_1}}$; if the first monitor appears after u (e.g., m_b), then $\mathcal{P}_l$ consists of l , $\mathcal{S}_{vi_3}$, $we_{11}u$, and $\mathcal{S}_{ui_2}$, and thus $W_l = W_{\mathcal{P}_l} - W_{\mathcal{S}_{vi_3}} - (W_{\mathcal{S}_{wi_1}} - W_{\mathcal{S}_{ui_1}}) - W_{\mathcal{S}_{ui_2}}$ (since the metric of $we_{11}u$ equals $W_{\mathcal{S}_{wi_1}} - W_{\mathcal{S}_{ui_1}}$). In all the cases, W_l can be computed in constant time.

The above complexity results are well aligned with needs in practice. Path construction only needs to be performed once for a given topology, and thus can tolerate a higher complexity. In contrast, link identification needs to be performed more fre-

quently to keep monitoring the health of links. In this regard, STLI in conjunction with STPC enables fast identification of link metrics that can scale to large networks.

4.5 An Example

Table 4.1: Paths for Identifying Tree Links

Node	Measurement Paths	Path Metrics
3 (m_4)	$\mathcal{P}_{v1} = 3\ 4\ 5\ 6\ 7 := \mathcal{P}^{(1)}$ $\mathcal{P}_{v2} = 3$ (omitted) $\mathcal{P}_{v3} = 3\ 9 := \mathcal{P}^{(2)}$	$W_{\mathcal{S}_{v1}} = W_{3\ 4\ 5\ 6\ 7}$ $W_{\mathcal{S}_{v2}} = 0$ $W_{\mathcal{S}_{v3}} = W_{3\ 9}$
4	$\mathcal{P}_{v1} = 7\ 6\ 5\ 4\ 3 := \mathcal{P}^{(3)}$ $\mathcal{P}_{v2} = 3\ 4\ 9 := \mathcal{P}^{(4)}$ $\mathcal{P}_{v3} = 9\ 4\ 5\ 6\ 7 := \mathcal{P}^{(5)}$	$W_{\mathcal{S}_{v1}} = W_{4\ 5\ 6\ 7}$ $W_{\mathcal{S}_{v2}} = W_{4\ 3}$ $W_{\mathcal{S}_{v3}} = W_{4\ 9}$
5	$\mathcal{P}_{v1} = 7\ 6\ 5\ 4\ 3 := \mathcal{P}^{(6)}$ $\mathcal{P}_{v2} = 3\ 4\ 5\ 9 := \mathcal{P}^{(7)}$ $\mathcal{P}_{v3} = 9\ 5\ 6\ 7 := \mathcal{P}^{(8)}$	$W_{\mathcal{S}_{v1}} = W_{5\ 6\ 7}$ $W_{\mathcal{S}_{v2}} = W_{5\ 4\ 3}$ $W_{\mathcal{S}_{v3}} = W_{5\ 9}$
6	$\mathcal{P}_{v1} = 7\ 6\ 5\ 4\ 3 := \mathcal{P}^{(9)}$ $\mathcal{P}_{v2} = 10\ 8\ 6\ 5\ 4\ 3 := \mathcal{P}^{(10)}$ $\mathcal{P}_{v3} = 10\ 8\ 6\ 7\ \mathcal{P}^{(11)}$	$W_{\mathcal{S}_{v1}} = W_{6\ 7}$ $W_{\mathcal{S}_{v2}} = W_{6\ 5\ 4\ 3}$ $W_{\mathcal{S}_{v3}} = W_{6\ 8\ 10}$
7 (m_2)	$\mathcal{P}_{v1} = 7$ (omitted) $\mathcal{P}_{v2} = 7\ 6\ 5\ 4\ 3 := \mathcal{P}^{(12)}$ $\mathcal{P}_{v3} = 7\ 8\ 10 := \mathcal{P}^{(13)}$	$W_{\mathcal{S}_{v1}} = 0$ $W_{\mathcal{S}_{v2}} = W_{7\ 6\ 5\ 4\ 3}$ $W_{\mathcal{S}_{v3}} = W_{7\ 8\ 10}$
8	$\mathcal{P}_{v1} = 7\ 8\ 6\ 5\ 4\ 3 := \mathcal{P}^{(14)}$ $\mathcal{P}_{v2} = 3\ 4\ 5\ 6\ 8\ 10 := \mathcal{P}^{(15)}$ $\mathcal{P}_{v3} = 10\ 8\ 7 := \mathcal{P}^{(16)}$	$W_{\mathcal{S}_{v1}} = W_{8\ 7}$ $W_{\mathcal{S}_{v2}} = W_{8\ 6\ 5\ 4\ 3}$ $W_{\mathcal{S}_{v3}} = W_{8\ 10}$
9 (m_3)	$\mathcal{P}_{v1} = 9\ 4\ 5\ 6\ 7 := \mathcal{P}^{(17)}$ $\mathcal{P}_{v2} = 9\ 3 := \mathcal{P}^{(18)}$ $\mathcal{P}_{v3} = 9$ (omitted)	$W_{\mathcal{S}_{v1}} = W_{9\ 4\ 5\ 6\ 7}$ $W_{\mathcal{S}_{v2}} = W_{9\ 3}$ $W_{\mathcal{S}_{v3}} = 0$
10 (m_1)	$\mathcal{P}_{v1} = 10\ 8\ 7 := \mathcal{P}^{(19)}$ $\mathcal{P}_{v2} = 10$ (omitted) $\mathcal{P}_{v3} = 10$ (omitted)	$W_{\mathcal{S}_{v1}} = W_{10\ 8\ 7}$ $W_{\mathcal{S}_{v2}} = 0$ $W_{\mathcal{S}_{v3}} = 0$

We illustrate the proposed algorithms with an example shown in Fig. 4.4 (a). It

can be verified that this network is identifiable under the given placement of four monitors m_j ($j = 1, \dots, 4$), i.e., it satisfies the condition in Theorem 3.7. STPC first constructs an extended graph $\mathcal{G}_{ex}^*$ as in Fig. 4.4 (b), where we label each node numerically for ease of presentation. It then uses the algorithm in [52] to find three spanning trees of $\mathcal{G}_{ex}^*$, as shown in Fig. 4.5, that are pairwise independent w.r.t. r . Based on these trees, STPC constructs three paths $\mathcal{P}_{vi}$ ($i = 1, 2, 3$) for each non-virtual node v (lines 4-11), of which all non-degenerate ones are considered measurement paths; see Table 4.1 for the results, where each path is denoted by a node sequence. After this step, the measurement matrix $\mathbf{R}$ consists of 19 monitor-to-monitor simple paths $\mathcal{P}^{(1)}\text{--}\mathcal{P}^{(19)}$, among which 9 are duplicates. Removing the duplicate paths yields 10 distinct measurement paths:

$$\begin{aligned}
& 3\ 4\ 5\ 6\ 7 \quad (\mathcal{P}^{(1)} = \mathcal{P}^{(3)} = \mathcal{P}^{(6)} = \mathcal{P}^{(9)} = \mathcal{P}^{(12)}) \\
& \quad 3\ 9 \quad (\mathcal{P}^{(2)} = \mathcal{P}^{(18)}) \\
& \quad 3\ 4\ 9 \quad (\mathcal{P}^{(4)}) \\
& 9\ 4\ 5\ 6\ 7 \quad (\mathcal{P}^{(5)} = \mathcal{P}^{(17)}) \\
& \quad 3\ 4\ 5\ 9 \quad (\mathcal{P}^{(7)}) \\
& \quad 9\ 5\ 6\ 7 \quad (\mathcal{P}^{(8)}) \\
& 10\ 8\ 6\ 5\ 4\ 3 \quad (\mathcal{P}^{(10)} = \mathcal{P}^{(15)}) \\
& \quad 10\ 8\ 6\ 7 \quad (\mathcal{P}^{(11)}) \\
& \quad 7\ 8\ 10 \quad (\mathcal{P}^{(13)} = \mathcal{P}^{(16)} = \mathcal{P}^{(19)}) \\
& \quad 7\ 8\ 6\ 5\ 4\ 3 \quad (\mathcal{P}^{(14)}).
\end{aligned} \tag{4.2}$$

Since there are three links (10 7, 7 5, and 5 3) not in any of the trees, STPC constructs one additional measurement path for each of these links using Algorithm 4.2 (lines 12-15), as shown in Table 4.2. Over all, STPC constructs 13 distinct measurement paths for the 13-link network in Fig. 4.4 (a).

After obtaining measurements on these paths, STLI first computes the node-to-monitor metrics for each node and each tree (lines 1-3), with results shown in

Table 4.2: Paths for Identifying Non-tree Links

Non-tree Link	Measurement Path
10 7	$\mathcal{P}_l = 10\ 7$
7 5	$\mathcal{P}_l = 7\ 5\ 9$
5 3	$\mathcal{P}_l = 7\ 6\ 5\ 3$

Table 4.1, where $W_{j_1 j_2 \dots j_k}$ denotes the sum metric of a path that goes through nodes $j_1, j_2, \dots, j_k$. Based on these results, 7 out of the 10 links in the trees are already identified (line 7); the other 3 links are identified as follows (line 9):

$$\begin{aligned}
\mathcal{T}_1 : W_{5\ 4} &= W_{4\ 5\ 6\ 7} - W_{5\ 6\ 7}, \\
W_{6\ 5} &= W_{5\ 6\ 7} - W_{6\ 7}, \\
\mathcal{T}_3 : W_{8\ 6} &= W_{6\ 8\ 10} - W_{8\ 10}.
\end{aligned} \tag{4.3}$$

Finally, the three links not on spanning trees are identified using known link metrics and the sum metrics of their corresponding measurement paths in Table 4.2.

4.6 Performance Evaluation

To evaluate the performance of STPC and STLI, we conduct a set of simulations on both randomly-generated and real network topologies. Given a network topology, the Minimum Monitor Placement (MMP) algorithm proposed in Section 6.1 (see Chapter 6 for details) is first applied to select a sub-set of nodes as monitors so that the network is guaranteed to be identifiable. We then apply the proposed algorithms, STPC and STLI, to construct measurement paths between the placed monitors and compute link metrics from measurements on these paths. The focus of our evaluations is on the *efficiency*, measured by the average running time, of the proposed algorithms in comparison to benchmarks. For path construction, we also evaluate the cost of measuring the constructed paths, in terms of average path length (i.e., number of hops).

As a benchmark for STPC, we use the following algorithm³, referred to as *Random Walk-based Path Construction (RWPC)*. Given an identifiable network $\mathcal{G}$, RWPC repeats the following steps until the rank of the constructed measurement matrix $\mathbf{R}$ equals n (starting from $\mathbf{R} = \emptyset$):

- (i) starting from a randomly selected monitor, follow a random walker until it hits another monitor;
- (ii) remove cycles from the path taken by the random walker to generate a simple monitor-to-monitor path;
- (iii) if the generated path is linearly independent w.r.t. existing paths in $\mathbf{R}$, append it to $\mathbf{R}$; otherwise, discard the path.

RWPC is essentially a randomized algorithm that examines one path at each iteration until n linearly independent paths are found. In practice, RWPC may iterate indefinitely for large networks. To control its running time, we impose a *maximum number of iterations* I_{MAX} , and force RWPC to terminate after I_{MAX} iterations. Consequently, we also measure its *success rate* r_{succ} , defined as the fraction of Monte Carlo runs during which RWPC successfully finds n linearly independent paths within I_{MAX} iterations (the success rate of STPC is always one). Limiting the number of iterations leads to underestimating the actual running time of RWPC in constructing n linearly independent paths, but it allows us to apply the algorithm to large networks.

As a benchmark for STLI, we use the general linear algebraic solution [7] that inverts the measurement matrix: $\mathbf{w} = \mathbf{R}^{-1}\mathbf{c}$, referred to as *Matrix Inversion-based Link Identification (MILI)*. Here $\mathbf{R}$ is an invertible matrix computed by RWPC if it is successful, or STPC otherwise.

Our simulation results include the following metrics:

- (a) r_{succ} : success rate of RWPC;

³Existing path-construction solutions cannot be used as benchmarks since they are not comparable to STPC for these two reasons: (i) most solutions assume given routing rather than controlled routing as assumed in this thesis, (ii) even under controlled routing assumption, existing path construction algorithms may contain cycles, which is prohibited in this thesis.

- (b) Υ : $\text{rank}(\mathbf{R})/n$ for RWPC when it is unsuccessful;
- (c) $t_{\text{STPC}}, t_{\text{RWPC}}$: average running times of STPC and RWPC;
- (d) $t_{\text{STLI}}, t_{\text{MILI}}$: average running times of STLI and MILI;
- (e) $h_{\text{STPC}}, h_{\text{RWPC}}$: average lengths of the n paths constructed by STPC and RWPC (when successful).

The simulation is implemented⁴ in Matlab R2010a and performed on a computer with Intel Core i5-2540M CPU @ 2.60GHz, 4.00 GB memory, and 64-bit Win7 OS.

4.6.1 Random Topologies

We first evaluate the proposed algorithms on synthetic topologies generated according to three different random graph models: Erdős-Rényi (ER) graphs, Random geometric (RG) graphs, and Barabási-Albert (BA) graphs. For each model, we fix the number of nodes to 150, and randomly generate 100 graph realizations⁵, which are then fed to the path construction algorithms. For each generated link, we randomly generate a link metric between 0 and 1, which is then used to compute path metrics. Here I_{MAX} is set to $3 \times n$. We now explain the models and the corresponding results separately.

4.6.1.1 Erdős-Rényi (ER) graph

The ER graph is a simple random graph 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. It is known [55] that $p_0 = \log |V|/|V|$ is a sharp threshold for the graph to be connected with high probability, which implies a minimum value of p ($p = 0.0334$ for $|V| = 150$).

⁴In this thesis, Matlab codes for basic graph operations are provided by [54].

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

4.6.1.2 Random geometric (RG) graph

The RG graph is frequently used to model the topology of wireless ad hoc networks. It generates a random graph by first randomly distributing nodes in a unit square, and then connecting each pair of nodes by a link if their distance is no larger than a threshold d_c , which denotes node communication range. The resulting topology contains well-connected sub-graphs in densely populated areas and poorly-connected sub-graphs in sparsely populated areas. It is known that $d_c \geq \sqrt{\log |V| / (\pi |V|)}$ ensures a connected graph with high probability [56], which gives a minimum range of $d_c = 0.1031$ for $|V| = 150$.

4.6.1.3 Barabási-Albert (BA) graphs

The BA model [57] provides a random power-law graph generated by the following preferential attachment mechanism. We begin with a small connected graph $\mathcal{G}_0 := (\{v_1, v_2, v_3, v_4\}, \{v_1v_2, v_1v_3, v_1v_4\})$ and add nodes sequentially. For each new node v , we connect v to $n_{\min}$ existing nodes, where $n_{\min}$ specifies (a lower bound on) the minimum node degree, such that the probability of connecting to node w is proportional to the degree of w . If the number of existing nodes is smaller than $n_{\min}$, then v connects to all existing nodes. The BA graph has been used to model many naturally occurring networks, e.g., Internet, citation networks, and social networks.

Simulation results are presented in Table 4.3, where each row corresponds to a random graph model, with results averaged over 100 graph realizations (recall that we have 100 graph realizations for each graph model). Since the number of links n vary across realizations, we present the average values denoted by $\bar{n}$. We have tuned parameters of each model to make the number of generated links roughly the same. In Table 4.3, most graphs are 3-vertex-connected, thus requiring only 3 monitors to achieve identifiability (see Theorem 3.7). From the results, we see that RWPC finds

Table 4.3: Random Graphs (ER: $p = 0.0656$, RG: $d_c = 0.15554$, BA: $n_{\min} = 5$, $I_{\max} = 3 \times n$)

graph	$\bar{n}$	$ V $	r_{succ}	Υ	t_{STPC} (s)	t_{RWPC} (s)	t_{STLI} (ms)	t_{MLI} (ms)	h_{STPC}	h_{RWPC}
ER	736.46	150	100.00%	NA	20.2	372.61	7.39	74.58	22.16	14.35
RG	739.57	150	28.00%	91.43%	20.49	918.42	8.09	74.16	29.16	21.44
BA	732	150	100.00%	NA	19.61	395.71	7.70	70.82	21.65	9.41

Table 4.4: ISP Topologies ($I_{\max} = 8 \times n$ for the first 5 networks, and $I_{\max} = \infty$ for the last 3)

ISP	n	$ V $	r_{succ}	Υ	t_{STPC} (s)	t_{RWPC} (s)	t_{STLI} (ms)	t_{MLI} (ms)	h_{STPC}	h_{RWPC}
Abovenet	294	182	80.00%	99.61%	10.12	58.20	2.46	5.08	5.68	4.03
EBONE	381	172	75.00%	99.69%	13.65	139.37	3.78	11.06	9.61	7.00
Tiscali	404	240	70.00%	99.67%	28.07	171.58	3.81	10.71	7.05	4.89
Exodus	434	201	67.00%	99.76%	21.13	226.15	4.13	14.49	8.26	6.13
Telstra	758	318	24.00%	99.76%	80.38	2999.96	6.70	118.17	7.86	6.22
AT&T	2078	631	NA	NA	685.46	131.1 hrs	19.50	1302.85	23.48	11.33
Sprintlink	2268	604	NA	NA	608.18	46.8 hrs	20.52	1560.55	15.03	11.06
Verio	2821	960	NA	NA	697.86	170.3 hrs	29.15	3366.79	13.22	8.97

all linearly independent paths successfully for ER and BA graphs, but fails most of the time for RG graphs. This is because node degrees vary significantly in RG graphs, and once the random walker hits a low-degree node, it has only a few paths to reach monitors, resulting in a high probability of generating duplicate paths. In fact, RWPC can quickly find a majority ($> 90\%$) of the linearly independent paths for RG graphs, but its efficiency drops sharply as the path set grows since most of the newly generated paths are linearly dependent with existing ones. In contrast, STPC only generates paths that are guaranteed to be useful in identifying additional links, thus significantly improving the efficiency. The improvement allows STPC to achieve a significantly smaller running time than RWPC, especially for RG graphs where we see a 45-fold speedup. Note that this is only an underestimate as RWPC often fails to find all the linearly independent paths for RG graphs, and the actual speedup is even bigger. Our link identification algorithm STLI also shows superior efficiency, reducing the running time of MILI by an order of magnitude. A further observation is that t_{STPC} , t_{STLI} , and t_{MILI} are roughly the same for different types of graphs, as their complexity are only determined by the size of the network (measured by n), whereas the running time t_{RWPC} is sensitive to the network connectivity. Meanwhile, we notice that STPC tends to generate paths that are longer than those generated by RWPC, especially for BA graphs. This is because STPC restricts paths to the spanning trees, selecting a longer path along spanning trees even if alternative shorter paths exist, while the random walker in RWPC is likely to take shorter paths to monitors. This is an intentional design in STPC to ensure linear independence of the constructed paths; it remains open as to whether the problem of minimizing path length while guaranteeing linear independence is NP-hard, which is left for future work.

4.6.2 ISP Topologies

We also test these algorithms on real network topologies. We use the *Internet Service Provider (ISP)* topologies from the Rocketfuel project [45], which represent physical connections between backbone/gateway routers of several major ISPs around the globe. The available data do not include link performance metrics; thus, we simulate link metrics by randomly generated numbers between 0 and 1. We point out that the performance of the algorithms is independent of the values of link metrics. Since RWPC is a randomized algorithm, we repeat it for multiple Monte Carlo runs for each ISP topology and report average performance; the number of Monte Carlo runs is 100 unless otherwise stated. In this simulation, we set $I_{\text{MAX}} = 8 \times n$ since we observe that $I_{\text{MAX}} = 3 \times n$ results in zero success rate for RWPC.

Simulation results are presented in Table 4.4, where we sort the networks according to their number of links (i.e., n). STPC again significantly outperforms RWPC with a speedup ranging from 6 fold (Abovenet, Tiscali) to 879 fold (Verio). In fact, RWPC becomes so slow for the largest three networks (AT&T, Sprintlink and Verio) that it is unable to complete a successful Monte Carlo run ($r_{\text{succ}} = 0\%$) even after 40 hours. To find out the time RWPC takes to find n linearly independent paths, we remove the limitation on the number of iterations ($I_{\text{MAX}} = \infty$) and let it run until success. RWPC takes up to 7 days (Verio) to complete a single Monte Carlo run⁶, which is in sharp contrast with STPC that finds n linearly independent paths in 10 minutes. For link identification, STLI also outperforms MILI with a speedup ranging from 2 fold (Abovenet) to 115 fold (Verio). Over all, we observe that the running-time advantages of STPC and STLI both increase with the size of the network, while the success rate and the efficiency of RWPC decay. As in the synthetic simulations, we again observe a relatively larger path length for STPC. However, the increase in path length is only moderate compared with the decrease in running time, and this is

⁶For this reason, we only conduct one Monte Carlo run for each of these three networks.

likely the cost needed to ensure linear independence of the paths.

Link Identification in Partially Identifiable Networks

If a network $\mathcal{G}$ with given monitor placement does not satisfy the complete identifiability condition in Theorem 3.7 (see Chapter 3), then it is impossible to identify all link metrics in $\mathcal{G}$. Nevertheless, sub-graphs of $\mathcal{G}$ may satisfy the conditions in Theorem 3.3 (see Chapter 3) or 3.7, and thus contain identifiable links. Since any forms of insights w.r.t. network internal states may assist network operators for fast localizing potential misbehavior, it is important to determine the set of identifiable links even if the network is not completely identifiable. This motivates the need for a finer-grained characterization of network identifiability in terms of the identifiability of individual links, i.e., partial network identifiability. However, there is no efficient method to determine which links are identifiable. For instance, although one can find out identifiable links by explicitly writing down the measurement matrix $\mathbf{R}$ based on *all* possible measurement paths and then finding the null space of $\mathbf{R}$ (links whose

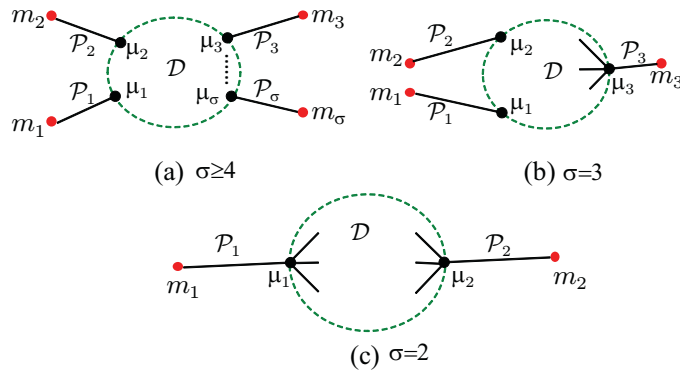


Figure 5.1: Three cases in identifying links of a 3-vertex-connected sub-graph $\mathcal{D}$.

corresponding entries are always zero in the basis of the null space are identifiable [9]), such an approach will incur exponential complexity due to the exponentially many measurement paths. In this chapter, we propose an alternative algorithm that leverages the fundamental identifiability conditions in Chapter 3 to efficiently find *all* the identifiable links in $\mathcal{G}$ (for a given monitor placement) in *linear* time.

5.1 Algorithm Overview

Our algorithm is based on two basic observations rooted in Theorems 3.3 and 3.7 in Chapter 3. For a 3-vertex-connected sub-graph $\mathcal{D}$ of $\mathcal{G}$:

Observation 1: if $\mathcal{D}$ contains or is connected via disjoint *external* (i.e., outside $\mathcal{D}$) links to three or more monitors, then $\mathcal{D}$ is completely identifiable since it can be verified that the supergraph of $\mathcal{D}$ including the monitors and their links to $\mathcal{D}$ satisfies the condition in Theorem 3.7;

Observation 2: if $\mathcal{D}$ is connected to two external monitors, each via at least two external links, then all the links in $\mathcal{D}$ are identifiable since it can be verified that the supergraph of $\mathcal{D}$ including the monitors and their links to $\mathcal{D}$ satisfies the conditions in Theorem 3.3; moreover, all the external links connected to the two monitors are unidentifiable by Corollary 3.2 (see Chapter 3).

Given an arbitrary network $\mathcal{G}$, if we decompose it into 3-vertex-connected sub-graphs (see Algorithm C.2 in Appendix C), then the above observation can be applied to determine link identifiability in each sub-graph. In general, $\mathcal{D}$ can be connected to monitors via multi-hop paths; for the sake of identifying links in $\mathcal{D}$, however, we will show that it suffices to only consider *internally vertex disjoint* paths, which are then modeled as logical links. Specifically, as illustrated in Fig. 5.1, $\mathcal{D}$ is connected to each monitor m_i from node μ_i via an internally vertex disjoint path $\mathcal{P}_i$. Note that m_i 's may contain duplicates to model multiple (internally vertex disjoint) connections to

the same monitor, while μ_i 's are all distinct. Intuitively, each μ_i provides a *vantage* (see Definition 11 for the formal definition) for monitoring the internal state of $\mathcal{D}$. We classify the possibilities in identifying $\mathcal{D}$ (assumed to be 3-vertex-connected) into three cases based on the number of vantages σ :

- (1) $\sigma \geq 4$ (Fig. 5.1 (a)): Abstracting each $\mathcal{P}_i$ as a single link, we see from Observation 1 that all links in $\mathcal{D}$ are identifiable if at least three of $m_1, \dots, m_\sigma$ are distinct; even if there are only two distinct monitors among $m_1, \dots, m_\sigma$, as long as each monitor is connected to at least two μ_i 's, we can still conclude the same from Observation 2.
- (2) $\sigma = 3$ (Fig. 5.1 (b)): In this case, if m_1, m_2 and m_3 are distinct, then we can apply Observation 1 to conclude that all links in $\mathcal{D}$ are identifiable; if two of m_1, m_2, m_3 are the same (e.g., $m_1 = m_2$), then Observation 2 applies, viewing $\mathcal{P}_1, \mathcal{P}_2$, and each concatenation of $\mathcal{P}_3$ and an adjacent link of μ_3 within $\mathcal{D}$ as single links, implying that all links in $\mathcal{D}$ except for those incident to μ_3 are identifiable (note that μ_3 has at least three links in $\mathcal{D}$ since $\mathcal{D}$ is 3-vertex-connected).
- (3) $\sigma = 2$ (Fig. 5.1 (c)): If m_1 and m_2 are distinct, then we can apply Observation 2, viewing each concatenation of $\mathcal{P}_1$ ($\mathcal{P}_2$) and an adjacent link of μ_1 (μ_2) as a single link, to conclude that all links in $\mathcal{D}$ except for those incident to μ_1 or μ_2 are identifiable.

We will show that these three cases represent all scenarios of interest in deter-

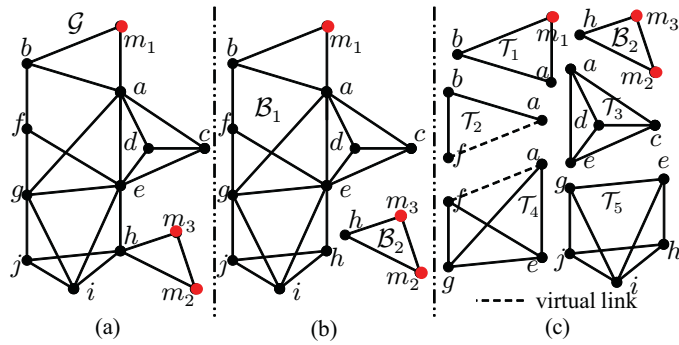


Figure 5.2: Decomposition of a sample network: (a) sample network $\mathcal{G}$; (b) biconnected components of $\mathcal{G}$; (c) triconnected components of $\mathcal{G}$.

mining link identifiability of a 3-vertex-connected sub-graph (e.g., the scenario of $\mathcal{D}$ connected to only one monitor is not of interest since none of its links can be probed via cycle-free paths); see Lemma 5.2 and Section 5.2.3. Our strategy for determining identifiable links in a general graph is to first decompose the graph into sub-graphs falling into one of the above cases, and then determine the set of identifiable links in each sub-graph.

5.2 Algorithm Details

5.2.1 Decomposition into Triconnected Components

The first step of our solution is to decompose the graph into suitably connected sub-graphs. To this end, we introduce the following definition.

Definition 8. 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 complete graph with up to k vertices. The case of $k = 2$ is also called a *biconnected component*, and $k = 3$ a *triconnected component*.

By this definition, a biconnected component can be either a 2-vertex-connected sub-graph or a single link (the case of an isolated vertex is excluded by assuming $\mathcal{G}$ to be connected as in the problem formulation of Chapter 2). Similarly, a triconnected component can be a 3-vertex-connected sub-graph, a triangle, or a single link. Intuitively, a biconnected component is a sub-graph connected to the rest of the graph by cut-vertices (see Definition 3 in Chapter 3), and a triconnected component (which always resides in a biconnected component) is a sub-graph connected to the rest of the biconnected component by 2-vertex cuts (see Definition 3 in Chapter 3). For instance, Fig. 5.2 (b) shows the biconnected components of Fig. 5.2 (a), separated by cut-vertex h . Fig. 5.2 (c) shows the triconnected components, separated by

cut-vertex h and 2-vertex cuts $\{a, b\}$, $\{a, f\}$, $\{a, e\}$, and $\{e, g\}$. However, not all sub-graphs separated by 2-vertex cuts form triconnected components according to Definition 8 (e.g., $\mathcal{T}_2$ in Fig. 5.2 (c)). To fix this issue, we process the graph to add *virtual links* as follows: for each 2-vertex cut whose vertices are not neighbors (e.g., $\{a, f\}$), connect these vertices by a virtual link; repeat this on the resulting graph until no such 2-vertex cut exists. It can be verified that all sub-graphs separated by cut-vertices or 2-vertex cuts in the processed graph are triconnected components.

There exist fast algorithms to decompose an arbitrary graph $\mathcal{G}$ into biconnected components [58] and then triconnected components [51]. Note that the output of [51] is the set of triconnected components of the processed graph¹.

5.2.2 Classification of Triconnected Components

After the decomposition, the next step is to determine to which case of Fig. 5.1 each triconnected component belongs. To this end, we introduce an auxiliary algorithm (Algorithm 5.1), which builds upon the graph decomposition algorithms [58, 51] (see Algorithms C.1 and C.2 in Appendix C) to classify the triconnected components according to their connections to monitors. Intuitively, nodes on the boundary of a triconnected component, i.e., cut-vertices and nodes in 2-vertex cuts, play crucial roles in determining its connections to monitors. The classification is therefore based on a few notions about these boundary nodes as introduced in Definitions 9–11 below.

The first notion, *agent*, relaxes the notion of monitor to also include nodes that effectively act as monitors.

Definition 9. An *agent* w.r.t. a sub-graph $\mathcal{G}'$ is either a monitor in $\mathcal{G}'$ or a cut-vertex that separates $\mathcal{G}'$ from at least one monitor.

¹The algorithm in [51] does not require preprocessing of adding virtual links. Virtual links are added as needed during graph partitioning.

Algorithm 5.1: Classification of Triconnected Components

input : Connected graph $\mathcal{G}$ with given monitor placement
output: Triconnected components and their categories

- 1 partition $\mathcal{G}$ into biconnected components $\mathcal{B}_1, \mathcal{B}_2, \dots$ [58] (see Algorithm C.1 in Appendix C) and identify agents for each biconnected component;
- 2 **foreach** biconnected component $\mathcal{B}_i$ with 2 or more agents **do**
- 3 partition $\mathcal{B}_i$ into triconnected components $\mathcal{T}_1, \mathcal{T}_2, \dots$ [51] (see Algorithm C.2 in Appendix C);
- 4 **foreach** triconnected component $\mathcal{T}_j$ of $\mathcal{B}_i$ **do**
- 5 identify vantages of $\mathcal{T}_j$ to determine its category;
- 6 **if** $|\mathcal{T}_j| = 3$ **then**
- 7 record all neighboring triconnected components;
- 8 **end**
- 9 **end**
- 10 **end**

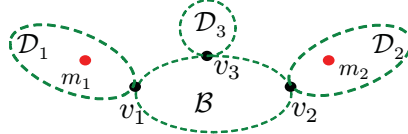
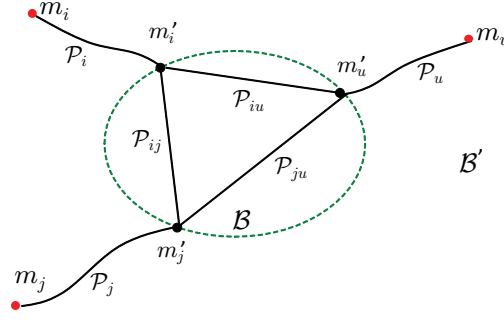
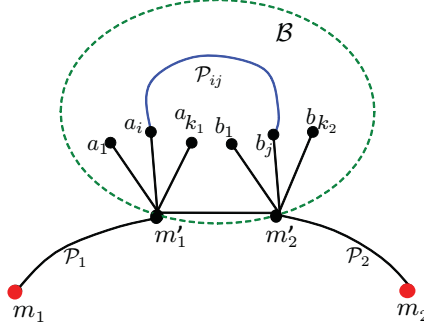


Figure 5.3: Agent: v_1 and v_2 are agents w.r.t. $\mathcal{B}$, but v_3 is not an agent w.r.t. $\mathcal{B}$ (m_1, m_2 : monitors).

We illustrate the concept of an agent in Fig. 5.3. Consider a sub-graph $\mathcal{B}$ illustrated in Fig. 5.3, which is separated from sub-graphs $\mathcal{D}_j$ by cut-vertices v_j ($j = 1, 2, 3$). We say that v_1 and v_2 are agents w.r.t. $\mathcal{B}$ since $\mathcal{D}_1$ and $\mathcal{D}_2$ both contain monitors; v_3 is not an agent w.r.t. $\mathcal{B}$ as $\mathcal{D}_3$ does not contain any monitor (but it is an agent w.r.t. $\mathcal{D}_3$). The significance of agents is that if $\mathcal{B}$ is a biconnected component, they essentially serve as “monitors” for identifying links in $\mathcal{B}$ as stated below.

Lemma 5.1. Let $\mathcal{B}$ be a biconnected component with agents $m'_1, \dots, m'_\kappa$. The identifiability of links in $\mathcal{B}$ does not depend on whether $m'_1, \dots, m'_\kappa$ are monitors or not, except for link $m'_1 m'_2$ (if it exists) when $\kappa = 2$.

Proof. 1) Consider the case with $\kappa \geq 3$: Suppose biconnected component $\mathcal{B}$ in Fig. 5.4 contains 3 or more agents. For any simple path $\mathcal{P}_{ij}$ connecting two agents m'_i and m'_j within $\mathcal{B}$, i.e., $V(\mathcal{P}_{ij}) \in V(\mathcal{B})$, it suffices to show that path metric $W_{\mathcal{P}_{ij}}$ can be calculated by path measurements between real monitors. Employing nodes

Figure 5.4: Biconnected component $\mathcal{B}$ with 3 or more agents.Figure 5.5: Agents m'_1 and m'_2 wrt biconnected component $\mathcal{B}$.

within $\mathcal{B}$, there exist two internally vertex disjoint paths $\mathcal{P}_{iu}$ and $\mathcal{P}_{ju}$ connecting m'_i and m'_j to another agent m'_u (the total agent number ≥ 3) with $V(\mathcal{P}_{ij} \cap \mathcal{P}_{iu}) = \{m'_i\}$ and $V(\mathcal{P}_{ij} \cap \mathcal{P}_{ju}) = \{m'_j\}$. This is because if any two of these three paths must have a common node (except for the common end-point), then this common node is a cut-vertex, contradicting the property of $\mathcal{B}$ being 2-vertex-connected. In addition, there exist three vertex disjoint paths $\mathcal{P}_i$, $\mathcal{P}_j$ and $\mathcal{P}_u$, each connecting an agent and a real monitor (see Fig. 5.4). Abstracting $\mathcal{P}_i$, $\mathcal{P}_j$, $\mathcal{P}_u$, $\mathcal{P}_{ij}$, $\mathcal{P}_{iu}$ and $\mathcal{P}_{ju}$ as single links, the augmented graph $\mathcal{B}'$ containing these six links and six nodes (m'_i , m'_j , m'_u , m_i , m_j , and m_u) satisfies the condition in Theorem 3.7, and thus $\mathcal{B}'$ is fully identifiable. Therefore, path metric of $W_{\mathcal{P}_{ij}}$ can be calculated by path measurements between real monitors when $\kappa \geq 3$.

2) Consider the case with $\kappa = 2$: Let m'_1 and m'_2 be the two agents of biconnected component $\mathcal{B}$ in Fig. 5.5 and m'_1 (m'_2) connects to the real monitor m_1 (m_2)

by path² $\mathcal{P}_1$ ($\mathcal{P}_2$), i.e., none of m'_1 and m'_2 are real monitors. In Fig. 5.5, it is impossible that $\mathcal{P}_1$ and $\mathcal{P}_2$ must have a common node; since otherwise m'_1 and m'_2 are not cut-vertices, contradicting the processing of localizing agents for a biconnected component. To identify link metrics in $\mathcal{B}$, all measurement paths involving links in $\mathcal{B}$ are of the following form

$$W_{\mathcal{P}_1} + W_{m'_1 a_i} + W_{\mathcal{P}_{ij}} + W_{b_j m'_2} + W_{\mathcal{P}_2} = c'_{ij}, \quad (5.1)$$

assuming $\mathcal{P}_1$ ($\mathcal{P}_2$) is always selected to connect m_1 and m'_1 (m_2 and m'_2). We know that if m'_1 and m'_2 are real monitors, then each measurement (except direct link $m'_1 m'_2$) path is of form

$$W_{m'_1 a_i} + W_{\mathcal{P}_{ij}} + W_{b_j m'_2} = c_{ij}. \quad (5.2)$$

Therefore, compared with (5.2), (5.1) is equivalent to abstracting each of $\mathcal{P}_1 + m'_1 a_i$ and $b_j m'_2 + \mathcal{P}_2$ as a single link. By Theorem 3.2, we know that none of the exterior links are identifiable. Thus, the link metrics of exterior links do not affect the identification of interior links. Therefore, $\mathcal{B}$ can be visualized as a network with two monitors m'_1 and m'_2 but each exterior link in $\{\{m'_1 a_i\}, \{m'_2 b_j\}\}$ has an added weight from $W_{\mathcal{P}_1}$ or $W_{\mathcal{P}_2}$. The above argument also holds when m_1 (m_2) chooses another path, say $\mathcal{P}'_1$ ($\mathcal{P}'_2$), to connect to m'_1 (m'_2), then it simply implies that different exterior links in $\{\{m'_1 a_i\}, \{m'_2 b_j\}\}$ in $\mathcal{B}$ may have different added path weights when regarding m'_1 and m'_2 as two monitors. Moreover, the above conclusion also applies to the case that one of m'_1 and m'_2 is a real monitor. Therefore, the identifiability of all links except for the direct link $l_d = m'_1 m'_2$ (if any) remains the same regardless of whether m'_1, m'_2 are monitors or not. $\square$

Remark: Since each agent is associated with at least one monitor, the number of agents w.r.t. a biconnected component cannot exceed the total number of monitors.

² $\mathcal{P}_1$ and $\mathcal{P}_2$ may not be unique.

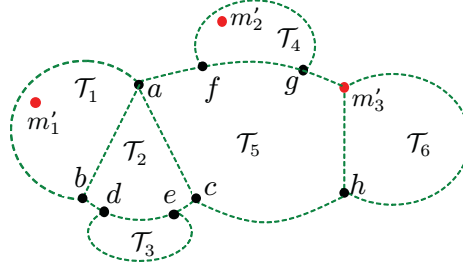


Figure 5.6: Type- k vertex cut: $\mathcal{T}_1, \dots, \mathcal{T}_6$ are triconnected components within the same biconnected component, for which m'_1, m'_2, m'_3 are agents.

Clearly, for any biconnected component with fewer than two agents, none of its links is identifiable since any monitor-to-monitor path going into this component via a node must leave through the same node, creating a cycle. Therefore, it suffices to only consider biconnected components with two or more agents.

The second notion, *type- k vertex cut*, classifies 2-vertex cuts based on the number of (effective) monitors behind a cut.

Definition 10. A *type- k vertex cut* (denoted by Type- k -VC) w.r.t. a triconnected component $\mathcal{T}$ is a 2-vertex cut $\{v_1, v_2\}$ in $\mathcal{T}$ that separates $\mathcal{T}$ from k agents (*excluding* agents in $\{v_1, v_2\}$) in its parent biconnected component (i.e., the biconnected component containing $\mathcal{T}$).

We illustrate the concept of a type- k vertex cut in Fig. 5.6, which contains a biconnected component consisting of six triconnected components and three agents. Each triconnected component $\mathcal{T}_i$ in Fig. 5.6 is separated from the rest by a set of 2-vertex cuts, each cut separating $\mathcal{T}_i$ from $k \in \{0, 1, 2, 3\}$ agents. The value of k determines the type of this cut. Specifically, in Fig. 5.6, we have: (i) Type-2-VC $\{a, b\}$ w.r.t. $\mathcal{T}_1$, (ii) Type-1-VC $\{a, b\}$, Type-2-VC $\{a, c\}$ and Type-0-VC $\{d, e\}$ w.r.t. $\mathcal{T}_2$, (iii) Type-3-VC $\{d, e\}$ w.r.t. $\mathcal{T}_3$, (iv) Type-2-VC $\{f, g\}$ w.r.t. $\mathcal{T}_4$, (v) Type-1-VC $\{f, g\}$, Type-0-VC $\{m'_3, h\}$ and Type-1-VC $\{a, c\}$ w.r.t. $\mathcal{T}_5$, and (vi) Type-2-VC $\{m'_3, h\}$ w.r.t. $\mathcal{T}_6$.

The third notion, *vantage*, further relaxes the notion of agent to also include

nodes in 2-vertex cuts with connections to monitors.

Definition 11. A *vantage* (denoted by μ) w.r.t. a triconnected component $\mathcal{T}$ is either an agent w.r.t. $\mathcal{T}$ or a node in a Type- k -VC w.r.t. $\mathcal{T}$ with $k \geq 1$.

Vantages w.r.t. a triconnected component $\mathcal{T}$ are the nodes in $\mathcal{T}$ that are connected to monitors, or equivalently agents, through *external paths* (i.e., paths not involving links in $\mathcal{T}$, including the degenerate path if the vantage is also an agent). For example, if the sub-graph $\mathcal{D}$ in Fig. 5.1 is a triconnected component, then all the nodes labeled μ_i ($i = 1, 2, \dots$) are vantages w.r.t. $\mathcal{D}$. We use Fig. 5.6 to illustrate the concept of vantages. According to Definition 11, the vantages for different triconnected components are (i) a, b, m'_1 for $\mathcal{T}_1$; (ii) a, b, c for $\mathcal{T}_2$; (iii) d, e for $\mathcal{T}_3$; (iv) f, g, m'_2 for $\mathcal{T}_4$; (v) a, c, f, g, m'_3 for $\mathcal{T}_5$; and (vi) h, m'_3 for $\mathcal{T}_6$.

Based on the above definitions, Algorithm 5.1 classifies all triconnected components of $\mathcal{G}$ according to the numbers of vantages that they contain. It first decomposes $\mathcal{G}$ into biconnected components using the algorithm in [58] and identifies the agents in each biconnected component (line 1). It then further decomposes each biconnected component of interest (i.e., with two or more agents) into triconnected components using the algorithm in [51] (line 3). For each triconnected component, it identifies all the vantages (line 5) by first determining the types of its 2-vertex cuts (w.r.t. itself) according to Definition 10, and then applying Definition 11. Based on the number of vantages, Algorithm 5.1 classifies the triconnected component into one of the following three categories (line 5):

- (i) **Category 1:** containing 4 or more vantages (e.g., $\mathcal{T}_5$ in Fig. 5.6);
- (ii) **Category 2:** containing 3 vantages (e.g., $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_4$ in Fig. 5.6);
- (iii) **Category 3:** containing 2 vantages (e.g., $\mathcal{T}_3$ and $\mathcal{T}_6$ in Fig. 5.6).

If the triconnected component has only three nodes (i.e., triangles), additional information is recorded (line 7), the purpose of which will be explained later (see Remark in Section 5.2.3).

As explained later in Section 5.2.3, these categories are in 1-1 correspondence with the three cases introduced in Section 5.1 and illustrated in Fig. 5.1. The following lemma guarantees that the above classification is complete for all the non-trivial cases (recall that biconnected components with fewer than two agents cannot have any identifiable link as discussed after Lemma 5.1).

Lemma 5.2. Any triconnected component within a biconnected component with two or more agents belongs to either one of **Category 1**, **2**, or **3**.

Proof. It is easy to see that **Categories 1–3** are mutually exclusive and cover all possibilities except for the case containing 0 or 1 vantage. If $\mathcal{T}$ only has 0 or 1 vantage, then the parent biconnected component $\mathcal{B}$ can only have 0 or 1 agent located in $\mathcal{T}$, as any other agent outside $\mathcal{T}$ will imply at least two vantages in $\mathcal{T}$ (nodes in the 2-vertex cut separating $\mathcal{T}$ from the agent must be vantages). Thus, any triconnected component with the parent biconnected component containing 2 or more agents falls into one of Categories 1–3. $\square$

5.2.3 Determination of All Identifiable Links

Based on the classification of triconnected components, we propose Algorithm 5.2, *Determination of All Identifiable Links (DAIL)*, to determine all the identifiable links in $\mathcal{G}$ under a given monitor placement. DAIL sequentially determines the identifiable links in each triconnected component of interest (i.e., in a biconnected component with 2 or more agents) based on its category (lines 3-15). These steps are best understood by referring to the illustration in Fig. 5.1, where each m_i represents an agent.

Category 1 (processed by line 4) is equivalent to the case in Fig. 5.1 (a). In particular, Category 1 allows different vantages to connect to the same agent (i.e., $m_1, \dots, m_\sigma$ in Fig. 5.1 (a) may not be distinct), implying that all links in a Category 1 triconnected component are identifiable as long as the number of vantages is at least

Algorithm 5.2: Determination of All Identifiable Links (DAIL)

input : Connected graph $\mathcal{G}$ with a given monitor placement
output: All identifiable links in $\mathcal{G}$

- 1 decompose $\mathcal{G}$ into triconnected components³ and determine the category of each triconnected component by Algorithm 5.1;
- 2 **foreach** *triconnected component $\mathcal{T}_i$ within a biconnected component with 2 or more agents* **do**
- 3 **if** $\mathcal{T}_i$ *is of Category 1* **then**
- 4 all links in $\mathcal{T}_i$ are identifiable;
- 5 **else if** $\mathcal{T}_i$ *is of Category 2 and 3-vertex-connected* **then**
- 6 **if** *only one of the three vantages in $\mathcal{T}_i$ is an agent, the other two form a Type-1-VC, and there is no other Type- k -VC with $k \geq 1$ in $\mathcal{T}_i$* **then**
- 7 all links in $\mathcal{T}_i$ except for the ones incident to the agent are identifiable;
- 8 **else**
- 9 all links in $\mathcal{T}_i$ are identifiable;
- 10 **end**
- 11 **else if** $\mathcal{T}_i$ *is of Category 2 and $\mathcal{T}_i$ is a triangle* **then**
- 12 the identification of $\mathcal{T}_i$ is determined by Algorithm A.3 (see Appendix A);
- 13 **else** // $\mathcal{T}_i$ must be of Category 3
- 14 all links in $\mathcal{T}_i$ except for the ones incident to the two vantages (μ_1 and μ_2) are identifiable; if $\exists$ link $\mu_1\mu_2$, then the identifiability of $\mu_1\mu_2$ is determined by Algorithm 5.3 ;
- 15 **end**
- 16 **end**

four. This is because the 2-vertex-connectivity of the parent biconnected component ensures the existence (see Lemma D.4 in Appendix D) of pairwise internally vertex disjoint paths $\{\mathcal{P}_1, \dots, \mathcal{P}_\sigma\}$ as shown in Fig. 5.1 (a). Accordingly, abstracting all these paths as single links, the graph in Fig. 5.1 (a) always satisfies Theorem 3.7. Similarly, Category 2 (processed by lines 6–10 and 12) and Category 3 (processed by line 14) are equivalent to the cases in Fig. 5.1 (b) and Fig. 5.1 (c), respectively. In particular, for a 3-vertex-connected component $\mathcal{T}$ of Category 2, we have shown in the explanation of Fig. 5.1 (b) (Section 5.1) that the connection between $\mathcal{T}$ and (external) agents is crucial in determining the identifiability of $\mathcal{T}$, i.e., depending on the number of distinct agents, $\mathcal{T}$ can be partially or completely identifiable. In this regard, lines 7 and 9 in Algorithm 5.2 are used to capture these factors affecting the identifiability of $\mathcal{T}$ of Category 2.

³According to Theorem D.8 in Appendix D, all triconnected components that are 3-vertex-

Remark: Link identifiability in a triconnected component cannot be directly determined from our previous observations in some special cases as follows. First, we have assumed that all the links can be employed by measurement paths, whereas the virtual links added during triconnected component decomposition (see explanations after Definition 8) cannot be used for measurements. Moreover, we have indicated in Lemma 5.1 that the identifiability of the direct agent-to-agent link in a biconnected component (or the link between two vantages in a triconnected component of Category 3) cannot be determined based on the internal structure of the component. For the first issue, we have proved in the Appendix that link identifiability in most triconnected components remains the same in the presence of virtual links. The only exceptions are Category 2 triangle components, which are handled separately in line 12 (by calling Algorithm A.3 in Appendix A). To determine link identifiability of a (Category 2) triangle component $\mathcal{T}$, Algorithm A.3 requires its neighboring triconnected components (see Appendix A for explanations); therefore, line 7 in Algorithm 5.1 is needed to provide this information. For the second issue, we separately determine the identifiability of direct links in triconnected components of Category 3 using Algorithm 5.3, the correctness of which is guaranteed by Lemma D.7 in Appendix D.

For Algorithm 5.2, Corollary 3.2 (together with the observation that no link is identifiable in a biconnected component with fewer than two agents) guarantees that links not classified as identifiable by DAIL are unidentifiable, i.e., DAIL determines *all* the identifiable links. Thus, we have Theorem 5.3.

Theorem 5.3. Algorithm 5.2 (DAIL) can determine all identifiable links in $\mathcal{G}$ for a given monitor placement.

connected (i.e., containing 4 or more nodes) or consist of a single-link are unique over all triconnected component decompositions, but the triangle components may not be unique. Nevertheless, Theorem 5.3 guarantees that the set of identifiable links determined by Algorithm 5.2 does not depend on the specific decomposition used.

Algorithm 5.3: Determination of Direct Links in Components of Category 3

input : Triconnected component $\mathcal{T}$ of Category 3 with direct link $\mu_1\mu_2$, and the neighboring biconnected components connected to $\mathcal{T}$ through μ_1 or μ_2

output: Identifiability of link $\mu_1\mu_2$

- 1 **if** *there exist neighboring triconnected components connecting to $\mathcal{T}$ by 2-vertex cut $\{\mu_1, \mu_2\}$ AND μ_1 and μ_2 are not the (only two) vantages in one of these neighboring components* **then**
- 2 $\mu_1\mu_2$ is determined in neighboring triconnected components;
- 3 **else**
- 4 Let S_1^B (S_2^B) be the set of biconnected components containing μ_1 (μ_2) as the only common node with $\mathcal{T}$, and n_1^B (n_2^B) be the total number of agents (excluding μ_1 and μ_2) w.r.t. each biconnected component in S_1^B (S_2^B);
- 5 **if** (μ_1 is a real monitor **OR** $n_1^B \geq 2$) **AND** (μ_2 is a real monitor **OR** $n_2^B \geq 2$) **then**
- 6 $\mu_1\mu_2$ is identifiable;
- 7 **else**
- 8 $\mu_1\mu_2$ is unidentifiable;
- 9 **end**
- 10 **end**

Proof. See Section E.4 in Appendix E. □

5.2.4 Complexity Analysis

In Algorithm 5.1, partitioning $\mathcal{G}$ into biconnected components (line 1) and then triconnected components (line 3) takes $O(|V| + |L|)$ time [58, 51]. The classification of triconnected components (lines 4-9 of Algorithm 5.1) can be performed during the partitioning with a constant-factor increase in the complexity. Thus, Algorithm 5.1, line 1 of Algorithm 5.2 (DAIL), has $O(|V| + |L|)$ time complexity. In DAIL, the complexity of lines 3-15 is $O(|L(\mathcal{T}_i)|)$ for each triconnected component $\mathcal{T}_i$ (Algorithms A.3 and 5.3 in DAIL each takes $O(1)$ time). Therefore, the overall complexity of DAIL is $O(|V| + |L|)$.

5.2.5 Example

The sample network in Fig. 5.2 (a) has 2 biconnected components $\mathcal{B}_1$ and $\mathcal{B}_2$ (Fig. 5.2 (b)) with agents $\{m_1, h\}$ and $\{h, m_2, m_3\}$, respectively. There are 5 triconnected com-

ponents in $\mathcal{B}_1$ (Fig. 5.2 (c)), where $\{\mathcal{T}_4\}$, $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_5\}$, and $\{\mathcal{T}_3\}$ belong to Categories 1, 2, and 3, respectively. Moreover, $\mathcal{B}_2$ itself is a Category 2 triconnected component. Therefore, the identifiable links are $\{ab, bf, cd, ae, ag, fe, fg, eg, ei, gi, gj, ij\}$ in $\mathcal{B}_1$, and $\{hm_2, hm_3, m_2m_3\}$ in $\mathcal{B}_2$.

Algorithms for Optimal Monitor Placement

We have established topological conditions in Chapter 3 for testing network identifiability and efficient algorithms in Chapter 5 for determining all identifiable links in partially identifiable networks under given monitor placement. The next questions we want to answer are: (i) Given an arbitrary topology $\mathcal{G}$, what is the minimum number ($\kappa_{\min}$) of monitors needed and where should they be placed to identify all link metrics in $\mathcal{G}$? (ii) In cases where the network operator only has a limited budget of κ monitors ($\kappa < \kappa_{\min}$), i.e., cannot achieve complete identifiability using κ monitors, where should these κ -monitors be placed to identify the largest sub-set of link metrics? For the first question, one can enumerate all possible placements for 3, 4, $\dots$ monitors and test for identifiability until an identifiable placement is found. While for the second question, the optimal κ -monitor placement can be obtained by exhaustive enumeration of all monitor placements, with each placement being tested by DAIL (see Chapter 5) to determine the number of identifiable links. However, what we really want are efficient algorithms to achieve the same. This requires a deeper understanding of the structure of an identifiable graph. These deeper understanding can then form the basis to our efficient polynomial-time algorithms addressing these two questions for monitor placement.

To this end, in this chapter, we first propose an optimal monitor placement algorithm for achieving complete network identifiability using the minimum number of

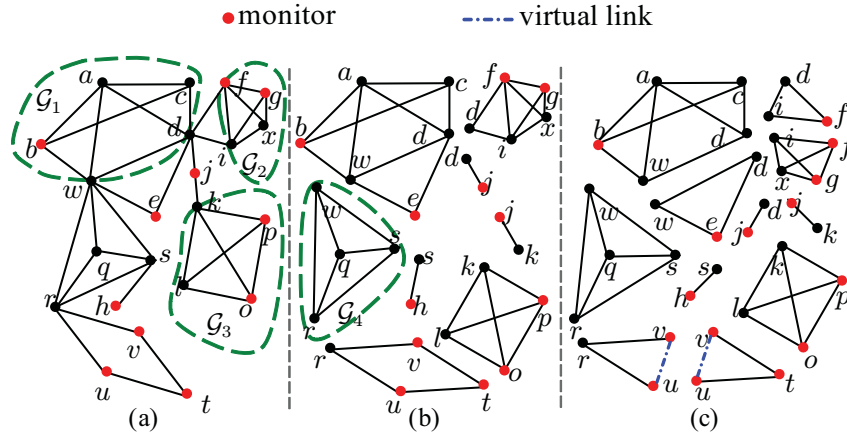


Figure 6.1: (a) Original graph; (b) biconnected components; (c) triconnected components.

monitors in Section 6.1 and then the efficient algorithms to maximize network identifiability in Section 6.2. Finally, the performance of these algorithms are evaluated in Section 6.3.

6.1 Optimal Monitor Placement for Complete Network Identifiability

The optimal monitor placement for ensuring complete network identifiability with the minimum number of monitors is motivated by the observation of minimum deployment: Using fewer monitors certainly renders the network unidentifiable. This yields the following rules, as illustrated in Fig. 6.1 (a):

- (i) *Dangling* node (e.g., h) must be a monitor, as otherwise its adjacent link cannot be measured using simple paths;
- (ii) A node on a *tandem* of links (e.g., j) must be a monitor, as otherwise we can only identify the sum of its adjacent link metrics (W_{dj} and W_{jk}) and not the individual metrics;
- (iii) For a sub-graph with *two cut-vertices* (e.g., G_1) or a *2-vertex cut* (e.g., G_2), at least one node other than those cuts must be a monitor, as otherwise even if all

Algorithm 6.1: Minimum Monitor Placement (MMP)

```

input : Connected graph  $\mathcal{G}$ 
output: A sub-set of nodes in  $\mathcal{G}$  as monitors
1 choose all the nodes with degree less than 3 as monitors;
2 partition  $\mathcal{G}$  into biconnected components  $\mathcal{B}_1, \mathcal{B}_2, \dots$ ;
3 foreach biconnected component  $\mathcal{B}_i$  with  $|\mathcal{B}_i| \geq 3$  do
4   partition  $\mathcal{B}_i$  into triconnected components  $\mathcal{T}_1, \mathcal{T}_2, \dots$ ;
5   foreach triconnected component  $\mathcal{T}_j$  of  $\mathcal{B}_i$  with  $|\mathcal{T}_j| \geq 3$  do
6     if  $0 < s_{\mathcal{T}_j} < 3$  and  $s_{\mathcal{T}_j} + M_{\mathcal{T}_j} < 3$  then
7       randomly choose  $3 - s_{\mathcal{T}_j} - M_{\mathcal{T}_j}$  nodes in  $\mathcal{T}_j$  that are neither separation
       vertices nor monitors as monitors;
8     end
9   end
10  if  $0 < c_{\mathcal{B}_i} < 3$  and  $c_{\mathcal{B}_i} + M_{\mathcal{B}_i} < 3$  then
11    randomly choose  $3 - c_{\mathcal{B}_i} - M_{\mathcal{B}_i}$  nodes in  $\mathcal{B}_i$  that are neither cut-vertices nor
    monitors as monitors;
12  end
13 end
14 if the total number of monitors  $\kappa < 3$  then
15   randomly choose  $3 - \kappa$  non-monitor nodes as monitors;
16 end

```

links outside this sub-graph have been identified, the vertices in the cuts (w and d for $\mathcal{G}_1$, f and i for $\mathcal{G}_2$) are effectively the two “monitors” for this sub-graph, and Theorem 3.1 in Chapter 3 states that this sub-graph cannot be completely identified;

- (iv) Similarly, for a sub-graph with *one cut-vertex* (e.g., $\mathcal{G}_3$), at least two nodes other than the cut-vertex must be monitors.

Our strategy is to use the above four rules to deploy the necessary monitors. If we can prove that these necessary monitors are also sufficient to identify all links, then our deployment is optimal.

6.1.1 Minimum Monitor Placement (MMP)

To formally present the algorithm, again, we decompose the given network into tri-connected components as introduced in Section 5.2.1 of Chapter 5. Using the fast algorithms for partitioning an arbitrary graph $\mathcal{G}$ into biconnected components [58]

and then into triconnected components [51] (see Algorithms C.1 and C.2 in Appendix C), we propose a master algorithm, *Minimum Monitor Placement (MMP)*, to place the minimum number of monitors needed to identify $\mathcal{G}$.

Definition 12. Nodes that are cut-vertices or part of 2-vertex cuts are called *separation vertices* (e.g., for the network in Fig. 6.1, cut-vertices d, j, k, w, s , and r and nodes in 2-vertex-cuts $\{w, d\}$, $\{f, i\}$, and $\{u, v\}$ are all separation vertices).

As shown in Algorithm 6.1, MMP first applies rules (i) and (ii) to select all the dangling vertices and vertices on tandems as monitors (line 1), and then applies rules (iii) and (iv) to select additional monitors in each¹ triconnected/biconnected component. For a component $\mathcal{D}$, let $s_{\mathcal{D}}$ denote the number of separation vertices, $c_{\mathcal{D}}$ the number of cut-vertices, and $M_{\mathcal{D}}$ the number of (already selected) monitors in $\mathcal{D}$. MMP goes through each triconnected and then biconnected component that contains three or more nodes to ensure that: (i) each triconnected component has at least three nodes that are either separation vertices or monitors (lines 6–8), and (ii) each biconnected component has at least three nodes that are either cut-vertices or monitors (lines 10–12). Finally, it selects additional monitors as needed to ensure that the total number of monitors is at least three (lines 14–16).

6.1.2 Optimality of MMP

It is easy to see from rules (i)–(iv) that MMP only deploys monitors when needed, and thus no algorithm can achieve identifiability with fewer monitors; on the other hand, we show that the monitor placement by MMP is also sufficient, i.e., all link metrics can be identified from end-to-end measurements between the selected monitors. Thus, MMP is optimal as it places the *minimum* number of monitors to identify all link metrics in $\mathcal{G}$, as stated in the following theorem.

¹MMP does not depend on the order of biconnected/triconnected components being considered for monitor selection.

Theorem 6.1. For an arbitrary connected network $\mathcal{G}$, Algorithm 6.1 (MMP) generates the *optimal* monitor placement in the sense that: (1) all link metrics in $\mathcal{G}$ are identifiable under this placement, and (2) no placement can identify all link metrics in $\mathcal{G}$ with a smaller number of monitors.

Proof. See Section E.5 in Appendix E. □

6.1.3 Complexity of MMP

In Algorithm 6.1, lines 1 and 14–16 take $O(|V|)$ time. Splitting $\mathcal{G}$ into biconnected (line 2) and then triconnected components (line 4) takes $O(|V| + |L|)$ time [58, 51]. Selecting monitors takes $O(1)$ time per component, and the counters ($s_{\mathcal{T}_j}$, $M_{\mathcal{T}_j}$, $c_{\mathcal{B}_i}$, and $M_{\mathcal{B}_i}$) can be computed during the splitting/selecting process. Therefore, the entire algorithm has $O(|V| + |L|)$ time complexity.

6.1.4 Example

Given the graph in Fig. 6.1 (a) as input (total of 22 nodes), MMP selects 11 monitors, where nodes $\{h, u, v, t, j, e\}$ are selected by line 1, $\{b, g, o, p\}$ by lines 6–8, and f by lines 10–12. It can be verified that the resulting graph satisfies the identifiability condition in Theorem 3.7.

Discussions on MMP: Compared with the works in [25, 37, 40] that show the minimum monitor placement for identifying all link metrics under *uncontrollable* routing is NP-hard, we prove that MMP is only in linear time complexity. This is because we assume all measurement paths are arbitrarily controllable, which immediately converts a generally NP-hard issue to be a problem that is linear time solvable. However, even in networks where measurement paths are not completely controllable, MMP is still significant in that it can yield a lower bound of the minimum monitor requirement for achieving network complete identifiability.

Algorithm 6.2: Candidate Monitor Selection

input : Triconnected components of $\mathcal{G}$ (computed by [51])
output: Set of candidates $\mathcal{S}$

```

1 foreach triconnected component  $\mathcal{T}$  of  $\mathcal{G}$  do
2   //  $\mathcal{A}^{\mathcal{T}}(v)$ : set of adjacent (non-virtual) links of node
    $v$  in  $\mathcal{T}$ ;
3   Under  $|\mathcal{A}^{\mathcal{T}}(v_1)| = \min_{v \in V(\mathcal{T})} |\mathcal{A}^{\mathcal{T}}(v)|$ , select  $v_1$  and  $v_2$  with the minimum
    $|\mathcal{A}^{\mathcal{T}}(v_1) \cup \mathcal{A}^{\mathcal{T}}(v_2)|$ ;
4   if the selection in line 3 is not unique then
5     Among alternatives generated by line 3, select  $\{v_1, v_2\}$  with the minimum
      $|\mathcal{A}^{\mathcal{T}}(v_1) \cup \mathcal{A}^{\mathcal{T}}(v_2) \setminus \{v_1 v_2\}|$  (break ties arbitrarily);
6   end
7   add  $\{v_1, v_2\}$  to  $\mathcal{S}$ ;
8   if  $|\mathcal{T}| \geq 3$  then
9     find  $v_3$  with the minimum  $|\mathcal{A}^{\mathcal{T}}(v)|$  over  $v \in V(\mathcal{T}) \setminus \{v_1, v_2\}$ ;
10    add  $v_3$  to  $\mathcal{S}$ ;
11  end
12 end

```

6.2 Maximal Identifiability Monitor Placement

In this section, we develop monitor placement algorithm in the case of limited resource for monitor deployment. Therefore, in the sequel, we assume $\kappa < \kappa_{\text{MMP}}$ in order to focus on the unresolved case of partially identifiable networks, where κ_{MMP} is the minimum number of monitors determined by MMP. In the proposed algorithm (Algorithm 6.3), we first identify candidates that can be part of an optimal monitor placement, and then develop an efficient algorithm to incrementally select monitors from the candidates. The significance of this algorithm is that it achieves optimality for a certain family of network topologies.

6.2.1 Candidate Monitor Selection

The first step of candidate selection is performed by Algorithm 6.2, based on the following argument. For a 3-vertex-connected network $\mathcal{G}$, Theorems 3.3 and Corollary 3.2 imply that, for a given 2-monitor placement, all links in $\mathcal{G}$ except for the links incident to the monitors are identifiable (note that the direct link connecting the

two monitors is always identifiable). Therefore, the optimal 2-monitor placement in $\mathcal{G}$ must be two nodes v_1 and v_2 with the minimum total number of adjacent links, excluding the direct link (as in lines 3 and 5 of Algorithm 6.2). For the same $\mathcal{G}$, if three monitors can be deployed, then $\mathcal{G}$ is completely identifiable according to Theorem 3.7, and no extra monitor is needed (although any placement of 3 monitors will achieve complete identifiability, the specific choice in line 9 is based on Theorem 6.2; see the proof in Appendix E). These observations inspire us to deploy monitors only in a sub-set $\mathcal{S}$ of nodes, known as *candidates*. The value of $\mathcal{S}$ is that it helps to reduce the set of nodes considered in monitor placement without losing optimality.

Theorem 6.2. The candidate set $\mathcal{S}$ selected by Algorithm 6.2 always contains an optimal monitor placement as a sub-set.

Proof. Let $d^{\mathcal{T}}(v) := |\mathcal{A}^{\mathcal{T}}(v)|$, $\mathcal{U}_{v_1 v_2} := \mathcal{A}^{\mathcal{T}}(v_1) \cup \mathcal{A}^{\mathcal{T}}(v_2)$, and $d_m^{\mathcal{T}} := \max d^{\mathcal{T}}(v)$, $v \in \{v_1, v_2, v_3\}$.

1) Suppose $|\mathcal{T}| \geq 3$. We first prove that $d^{\mathcal{T}}(w) \geq d_m^{\mathcal{T}}$ for $w \in V(\mathcal{T}) \setminus \{v_1, v_2, v_3\}$. According to lines 3 and 5 of Algorithm 6.2, we have $d^{\mathcal{T}}(v_1) + d^{\mathcal{T}}(v_2) - 1 \leq d^{\mathcal{T}}(v_1) + d^{\mathcal{T}}(v_3)$, and thus $d^{\mathcal{T}}(v_2) \leq d^{\mathcal{T}}(v_3) + 1$. There are two possible cases for the relationship between $d^{\mathcal{T}}(v_2)$ and $d^{\mathcal{T}}(v_3)$. (i) $d^{\mathcal{T}}(v_2) \geq d^{\mathcal{T}}(v_3)$: Then $d^{\mathcal{T}}(v_3) \leq d^{\mathcal{T}}(v_2) \leq d^{\mathcal{T}}(v_3) + 1$. Accordingly, we have $d^{\mathcal{T}}(v_2) - d^{\mathcal{T}}(v_3) = 1$ or $d^{\mathcal{T}}(v_2) = d^{\mathcal{T}}(v_3)$ as $d^{\mathcal{T}}(\cdot)$ is an integer. Hence, there is no integer $d^{\mathcal{T}}(w)$ with $d^{\mathcal{T}}(v_3) < d^{\mathcal{T}}(w) < d^{\mathcal{T}}(v_2)$. Moreover, based on the rule to select v_3 by line 9, there is no node w with $d^{\mathcal{T}}(v_1) < d^{\mathcal{T}}(w) < d^{\mathcal{T}}(v_3)$. (ii) $d^{\mathcal{T}}(v_2) \leq d^{\mathcal{T}}(v_3)$: Then $d^{\mathcal{T}}(v_1) \leq d^{\mathcal{T}}(v_2) \leq d^{\mathcal{T}}(v_3)$. If there exists $d^{\mathcal{T}}(w)$ with $d^{\mathcal{T}}(v_2) < d^{\mathcal{T}}(w) < d^{\mathcal{T}}(v_3)$, then v_3 is not chosen according to line 9. If there exists $d^{\mathcal{T}}(w)$ with $d^{\mathcal{T}}(v_1) < d^{\mathcal{T}}(w) < d^{\mathcal{T}}(v_2)$, then we should select this w as v_3 , thus contradicting the assumption that $d^{\mathcal{T}}(v_2) \leq d^{\mathcal{T}}(v_3)$. Therefore, $d^{\mathcal{T}}(w) \geq d_m^{\mathcal{T}}$ for $w \in V(\mathcal{T}) \setminus \{v_1, v_2, v_3\}$.

2) Now we prove the completeness of $\mathcal{S}$. It suffices to show that there exists optimal monitor placement for each triconnected component by using only the nodes

in $\mathcal{S}$. There are two possible scenarios: (i) $|\mathcal{T}| = 2$ or 3: All nodes in $\mathcal{T}$ are candidates. (ii) $|\mathcal{T}| \geq 4$: (a) If $\mathcal{T}$ is completely identifiable, then $\{v_1, v_2, v_3\}$ can all be selected as monitors and no additional nodes in $\mathcal{T}$ are required to be monitors for completely identifying $\mathcal{T}$; (b) If $\mathcal{T}$ is not completely identifiable, then according to the three categories of triconnected components, $\mathcal{T}$ must be of Category 2.2 (defined in Appendix A) or 3 (as $\mathcal{T}$ with $|\mathcal{T}| \geq 4$ is completely identifiable if it is of the other categories). When $\mathcal{T}$ is of Category 2.2, then only links incident to one vantage, e.g., μ_3 in Fig. 5.1 (b), in $\mathcal{T}$ are unidentifiable. For such case, as illustrated in Fig. 5.1 (b), we characterize the relationship between candidates and vantages $\{\mu_1, \mu_2\}$ by set $\Delta := \{v_1, v_2, v_3\} \setminus \{\mu_1, \mu_2\}$. Then we select a node in Δ with the minimum number of adjacent links within $\mathcal{T}$ as a monitor, e.g., μ_3 in Fig. 5.1 (b). According to the argument in 1), there is no node w with $d^{\mathcal{T}}(w) < d_m^{\mathcal{T}}$ for $w \in V(\mathcal{T}) \setminus \{v_1, v_2, v_3\}$. Therefore, for all nodes (in $\mathcal{T} \setminus \{\mu_1, \mu_2\}$) capable of ensuring $\mathcal{T}$ to be of Category 2.2, the one with the minimum number of adjacent links (actually can be chosen within Δ) is selected as a monitor, i.e., the optimal case for $\mathcal{T}$ of Category 2.2. Now consider the case that $\mathcal{T}$ with $|\mathcal{T}| \geq 4$ is of Category 3. In this case, if two monitors can be selected from $\mathcal{T}$, then v_1 and v_2 are chosen to be the two vantages in $\mathcal{T}$; therefore, the number of unidentifiable links in $\mathcal{T}$ is minimized (by lines 3 and 5), i.e., $|\mathcal{U}_{v_1 v_2}|$ (or $|\mathcal{U}_{v_1 v_2}| - 1$ when there exists a direct link $v_1 v_2$ and $v_1 v_2$ is identifiable by Algorithm 5.3). If there exists a vantage μ_0 and only one monitor can be chosen from $\mathcal{T}$, then $v = \arg \min_{v \in \{v_1, v_2, v_3\} \setminus \mu_0} d(v)$ is selected as a monitor. Therefore, the generated candidate set $\mathcal{S}$ is complete. $\square$

Let $N_{\mathcal{T}}$ denote the total number of triconnected components² in $\mathcal{G}$. Since each triconnected component contains at most three candidates, the total number of candidates in $\mathcal{S}$ ($|\mathcal{S}|$ can be much smaller than $|V|$) is upper bounded by $3N_{\mathcal{T}}$.

²Although the decomposition into triconnected components may not be unique, the total number of triconnected components $N_{\mathcal{T}}$ is always the same according to Theorem D.8 in Appendix D.

Algorithm 6.3: Greedy Maximal identifiability Monitor Placement (GMMP)

input : Connected graph $\mathcal{G}$, its candidate set $\mathcal{S}$
output: A κ -monitor placement $\mathcal{O}_\kappa$

```

1 if  $\kappa = 2$  then
2   | enumerate all pairs of nodes in  $\mathcal{S}$ , return  $\mathcal{O}_2 = \arg \max_{v_1, v_2 \in \mathcal{S}} N(\{v_1, v_2\})$ ;
3 else if  $\kappa \geq 3$  then
4   | enumerate all triples of nodes in  $\mathcal{S}$ , return
   |  $\mathcal{O}_3 = \arg \max_{v_1, v_2, v_3 \in \mathcal{S}} N(\{v_1, v_2, v_3\})$ ;
5   foreach  $i, i = \{4, \dots, \kappa\}$  do
6     | find  $v_m = \arg \max_v N(\mathcal{O}_{i-1} \cup \{v\})$  over  $v \in \mathcal{S} \setminus \mathcal{O}_{i-1}$ ;
7     | set  $\mathcal{O}_i = \mathcal{O}_{i-1} \cup \{v_m\}$ ;
8   end
9 end
10 return  $\mathcal{O}_\kappa$ ;

```

6.2.2 Greedy Maximal identifiability Monitor Placement (GMM-P)

Given an algorithm to calculate the number of identifiable links (using Algorithm 5.2 - DAIL) and preprocessing to identify candidate monitors (using Algorithm 6.2), we are ready to develop a monitor placement algorithm to maximize (partial) identifiability. While the optimal solution is combinatorial in nature and therefore likely to be computationally expensive, we propose a simple low-complexity algorithm that we show, both theoretically and empirically, to have satisfactory performance.

We propose an incremental greedy algorithm, *Greedy Maximal identifiability Monitor Placement (GMMP)*, that builds upon a suitably-constructed initial placement to select additional monitors, one at a time, so that each selection maximizes the gain in identifiability; see Algorithm 6.3, where $N(\mathcal{O})$ denotes the number of identifiable links under the monitor placement $\mathcal{O}$ (as defined in Mathematical Notations). The bulk of computation is in constructing this initial placement. GMMP constructs an optimal κ_0 -monitor placement for a small value of κ_0 ($\kappa_0 = 2$ for $\kappa = 2$, or 3 for $\kappa \geq 3$) by exhaustively enumerating all selections of κ_0 monitors from the candidate set $\mathcal{S}$ (lines 2, 4). If more monitors are available ($\kappa > 3$), it then iteratively selects

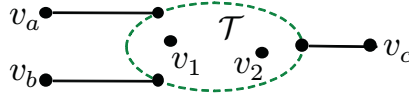


Figure 6.2: Example: $\mathcal{O}_2 = \{v_1, v_2\}$ is not a sub-set of $\mathcal{O}_3 = \{v_a, v_b, v_c\}$.

additional monitors so that each newly selected monitor maximizes the number of additional identifiable links (lines 5-8).

The reason for handling the cases of $\kappa = 2$ and $\kappa \geq 3$ separately is that the optimal 3-monitor placement $\mathcal{O}_3$ is not necessarily a superset of the optimal 2-monitor placement $\mathcal{O}_2$. For example, Fig. 6.2 is a graph consisting of a large triconnected component $\mathcal{T}$ and three dangling nodes connected to $\mathcal{T}$ via three cut-vertices. Then $\mathcal{O}_2$ can be two nodes $\{v_1, v_2\}$ in $\mathcal{T}$ with the minimum number of neighbors within $\mathcal{T}$, while $\mathcal{O}_3$ are the dangling nodes $\{v_a, v_b, v_c\}$ (which can be shown to identify all the links in this graph by Theorem 3.7).

Remark: Although we have used the *number* of identifiable links as the optimization criterion in GMMP, all the evaluated algorithms are applicable to a more general criterion given as an arbitrary mapping from the set of identifiable links to a utility value, as DAIL can actually determine the *set* of identifiable links. We leave performance analysis and evaluation in this generalized setting to future work.

6.2.3 Optimality of GMMP for 2-vertex-connected Networks

Besides being an efficient heuristic, GMMP also has theoretical significance in that it is optimal for a family of network topologies that are 2-vertex-connected. Consider a graph $\mathcal{G}$ that is 2-vertex-connected, i.e., containing no cut-vertex. Let $\mathcal{M}$ be a minimum set of monitors given by MMP³ for complete identification of $\mathcal{G}$. Let $\mathcal{O}_\kappa^*$ denote an optimal κ -monitor placement in $\mathcal{G}$ ($\mathcal{O}_\kappa^*$ may not be unique). We have the following theorem.

³This set may not be unique, but the number of nodes in $\mathcal{M}$, i.e., $|\mathcal{M}|$, is unique, and our argument in Theorem 6.3 holds for any such $\mathcal{M}$.

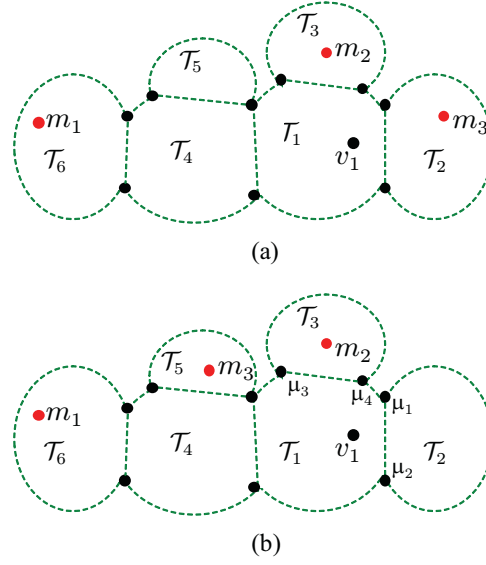


Figure 6.3: Optimal monitor placement in 2-vertex-connected networks.

Theorem 6.3. If $\mathcal{G}$ is 2-vertex-connected and $3 \leq \kappa < |\mathcal{M}|$, then (i) $\exists \mathcal{O}_\kappa^*$ such that $\mathcal{O}_\kappa^* \subseteq \mathcal{M}$, and (ii) for any given $\mathcal{O}_\kappa^*$ with $\mathcal{O}_\kappa^* \subseteq \mathcal{M}$, $\exists \mathcal{O}_{\kappa+1}^*$ such that $\mathcal{O}_{\kappa+1}^* = \mathcal{O}_\kappa^* \cup \{v_m\}$, where $v_m = \arg \max_v N(\mathcal{O}_\kappa^* \cup \{v\})$ over $v \in \mathcal{M} \setminus \mathcal{O}_\kappa^*$.

Proof. We prove it by contradiction.

Definition 13. A vantage v is *independent* if (i) it is an agent, or (ii) it belongs to a Type-1-VC $\{v, v'\}$ where v' is a monitor, or (iii) it belongs to a Type- k -VC with $k \geq 2$.

1) We first prove $\mathcal{O}_\kappa^* \subseteq \mathcal{M}$.

(1.i). When $|\mathcal{M}| = 3$, then it is trivial that $\mathcal{O}_3^* = \mathcal{M}$.

(1.ii). When $|\mathcal{M}| > 3$, as $\kappa \geq 3$ and $\mathcal{G}$ is 2-vertex-connected, each triconnected component involves at least 2 independent vantages. Therefore, all links in triconnected components (except for some links in triangles of Category 2) with 3 or more vantages are identifiable, and only links incident to the two vantages in the triconnected components of Category 3 are unidentifiable. Suppose $\mathcal{O}_\kappa^*$ with $\mathcal{O}_\kappa^* \subseteq \mathcal{M}$ does not exist, then it implies that some nodes in $\mathcal{O}_\kappa^*$ cannot be selected from $\mathcal{M}$. In this case, at least one node in $\mathcal{O}_\kappa^*$ is within a triconnected component with 3 or more

separation nodes (see Definition 12), say v_1 in Fig. 6.3, since this location cannot be selected by MMP when $|\mathcal{M}| > 3$. If v_1 is in the location illustrated as Fig. 6.3 (a), i.e., all neighboring triconnected components of $\mathcal{T}_1$ within $\mathcal{G}$ contain monitors, then placing a monitor at v_1 does not contribute link identification in $\mathcal{G}$; therefore, placing a monitor at v_1 is not the optimal solution. Now suppose v_1 is in the location illustrated as Fig. 6.3 (b), i.e., at least one neighboring triconnected component ($\mathcal{T}_2$) of $\mathcal{T}_1$ contains no monitors. In this case, placing a monitor at v_1 does not contribute link identification in $\mathcal{G}$ either, except for identifying the links in triangles (if $\mathcal{T}_1$ is a triangle, see explanations in Algorithm A.3). Nevertheless, moving the monitor from v_1 to a node⁴ (a node other than μ_1 and μ_2) in $\mathcal{T}_2$ can maintain the links which are identifiable when v_1 is a monitor. Moreover, links incident to μ_1 and μ_2 in $\mathcal{T}_2$ become identifiable under this new monitor location, which can be selected by MMP, i.e., the new location forms a better placement (which can be selected within $\mathcal{M}$) than $\mathcal{O}_\kappa^*$, contradicting the assumption that $\mathcal{O}_\kappa^*$ containing nodes outside $\mathcal{M}$ is the optimal κ -monitor placement.

Hence, based on (i) and (ii), there exists optimal solution $\mathcal{O}_\kappa^*$ with $\mathcal{O}_\kappa^* \subseteq \mathcal{M}$.

2) Now we prove $\mathcal{O}_{\kappa+1}^*$ can be constructed by $\mathcal{O}_{\kappa+1}^* = \mathcal{O}_\kappa^* \cup \{v_m\}$, where $v_m \in \mathcal{M}$, $\mathcal{O}_\kappa^* \subseteq \mathcal{M}$.

We obtain node set $\mathcal{V}_{\kappa+1}$ by adding node $v_{\kappa+1}$ to set $\mathcal{O}_\kappa^*$ such that $N(\mathcal{V}_{\kappa+1})$ is maximized. Then on top of $\mathcal{O}_\kappa^*$, suppose $|\alpha_1| + |\alpha_2|$ extra identifiable links are achieved by adding node $v_{\kappa+1}$, i.e., $|\alpha_1| + |\alpha_2| = N(\mathcal{V}_{\kappa+1}) - N(\mathcal{O}_\kappa^*)$, where α_1 is the set of effective exterior links in the triconnected component (e.g., $\mathcal{T}_1$ in Fig. 6.4 (a)) involving $v_{\kappa+1}$ when nodes in $\mathcal{O}_\kappa^*$ are employed as monitors, and α_2 is the set of all other identifiable links (e.g., effective exterior links in $\mathcal{T}_2$ and $\mathcal{T}_3$ in Fig. 6.4 (a)) determined by adding $v_{\kappa+1}$. Now suppose $N(\mathcal{V}_{\kappa+1}) < N(\mathcal{O}_{\kappa+1}^*)$, then moving nodes in $\mathcal{V}_{\kappa+1}$ to specific locations, i.e., $\mathcal{V}'_{\kappa+1}$, can get $N(\mathcal{V}'_{\kappa+1}) > N(\mathcal{V}_{\kappa+1})$. For $\mathcal{V}'_{\kappa+1}$,

⁴Theorem 6.3 only applies to the case that $\kappa \geq 3$, since the following properties cannot be guaranteed if $\kappa = 2$ in $\mathcal{G}$.

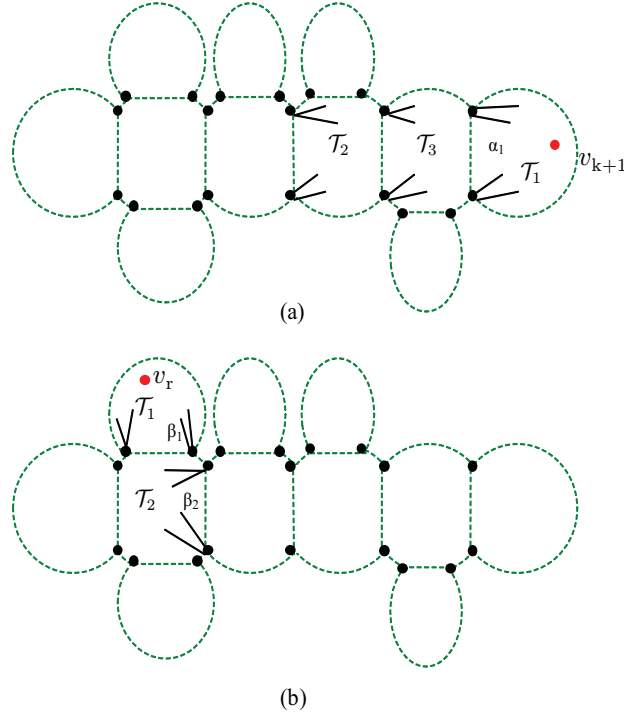


Figure 6.4: Nested structure of optimal monitor placement in 2-vertex-connected networks.

there are two possible cases.

(2.i). $v_{\kappa+1} \in \mathcal{V}'_{\kappa+1}$: In $\mathcal{V}'_{\kappa+1}$, let $\mathcal{V}_\kappa := \mathcal{V}'_{\kappa+1} \setminus v_{\kappa+1}$. We have 3 cases for $\mathcal{V}_\kappa$, denoted by $\mathcal{V}_\kappa^{(1)}$, $\mathcal{V}_\kappa^{(2)}$, and $\mathcal{V}_\kappa^{(3)}$. If $\mathcal{V}_\kappa^{(1)}$ can determine the identification of all links in α_2 , then

$$N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_\kappa^{(1)}) + |\alpha_1|; \quad (6.1)$$

if $\mathcal{V}_\kappa^{(2)}$ can only determine the identification of some links in α_2 , then

$$N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_\kappa^{(2)}) + |\alpha_i|, \quad (6.2)$$

where $|\alpha_i|$ is an integer with $|\alpha_1| \leq |\alpha_i| \leq |\alpha_1| + |\alpha_2|$; finally, if $\mathcal{V}_\kappa^{(3)}$ cannot determine the identification of any links in α_1 and α_2 , then we have

$$N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_\kappa^{(3)}) + |\alpha_1| + |\alpha_2|. \quad (6.3)$$

We know that $N(\mathcal{V}_\kappa^{(1)}) \leq N(\mathcal{O}_\kappa^*)$, $N(\mathcal{V}_\kappa^{(2)}) \leq N(\mathcal{O}_\kappa^*)$, and $N(\mathcal{V}_\kappa^{(3)}) \leq N(\mathcal{O}_\kappa^*)$. Moreover, according to the way of getting $\mathcal{V}_{\kappa+1}$, it is achievable for $N(\mathcal{V}_\kappa^{(3)})$ to

be $N(\mathcal{O}_\kappa^*)$. Therefore, the best case for $\mathcal{V}'_{\kappa+1}$ is that $N(\mathcal{V}_\kappa^{(3)}) = N(\mathcal{O}_\kappa^*)$ in (6.3), which implies $N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_{\kappa+1})$, contradicting the assumption that $N(\mathcal{V}'_{\kappa+1}) > N(\mathcal{V}_{\kappa+1})$.

(2.ii). $v_{\kappa+1} \notin \mathcal{V}'_{\kappa+1}$: In this case, there exists node v_r with $v_r \notin \mathcal{O}_\kappa^*$ in $\mathcal{V}'_{\kappa+1}$. Accordingly, $\mathcal{V}'_{\kappa+1}$ can be written in the form of $\mathcal{V}'_{\kappa+1} = \mathcal{V}_\kappa \cup v_r$. Then on top of $\mathcal{V}_\kappa$, suppose $|\beta_1| + |\beta_2|$ extra identifiable links are achieved by adding node v_r , where β_1 is the set of effective exterior links in the triconnected component (e.g., $\mathcal{T}_1$ in Fig. 6.4 (b)) involving v_r when nodes in $\mathcal{V}_\kappa$ are employed as monitors, and β_2 is the set of all other identifiable links (e.g., effective exterior links in $\mathcal{T}_2$ in Fig. 6.4 (b)) determined by adding v_r . Then moving nodes in $\mathcal{V}_\kappa$ to other locations, i.e., $\mathcal{V}'_\kappa$, there are three possible cases, denoted by $\mathcal{V}_\kappa^{(1)}$, $\mathcal{V}_\kappa^{(2)}$, and $\mathcal{V}_\kappa^{(3)}$. If $\mathcal{V}_\kappa^{(1)}$ can determine the identification of all links in β_2 , then

$$N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_\kappa^{(1)}) + |\beta_1|; \quad (6.4)$$

if $\mathcal{V}_\kappa^{(2)}$ can only determine the identification of some links in β_2 , then

$$N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_\kappa^{(2)}) + |\beta_i|, \quad (6.5)$$

where $|\beta_i|$ is an integer with $|\beta_1| \leq |\beta_i| \leq |\beta_1| + |\beta_2|$; finally, if $\mathcal{V}_\kappa^{(3)}$ cannot determine the identification of any links in β_1 and β_2 , then we have

$$N(\mathcal{V}'_{\kappa+1}) = N(\mathcal{V}_\kappa^{(3)}) + |\beta_1| + |\beta_2|. \quad (6.6)$$

Note that the correctness of (6.4)–(6.6) are ensured by the fact that $\kappa \geq 3$. Following the similar argument in (2.i), since $v_r \notin \mathcal{O}_\kappa^*$, it is achievable for $N(\mathcal{V}_\kappa^{(3)})$ to be $N(\mathcal{O}_\kappa^*)$ in (6.6), which implies $\mathcal{V}'_{\kappa+1} = \mathcal{O}_\kappa^* \cup v_r$. We know $v_r \neq v_{\kappa+1}$, $v_{\kappa+1} \notin \mathcal{O}_\kappa^*$, and $N(\mathcal{O}_\kappa^* \cup v_r) \leq N(\mathcal{O}_\kappa^* \cup v_{\kappa+1})$; therefore, even for the best case of $\mathcal{V}'_{\kappa+1}$ with $v_{\kappa+1} \notin \mathcal{V}'_{\kappa+1}$, it cannot achieve a larger number of identifiable links than that determined by $\mathcal{O}_\kappa^* \cup v_{\kappa+1}$. Thus, no matter how we move the nodes in $\mathcal{V}_{\kappa+1}$ to other locations, it

is impossible to find a monitor placement achieving a larger number of identifiable links. Therefore, $\mathcal{V}_{\kappa+1} = \mathcal{O}_{\kappa+1}^*$.

Consequently, the optimal $(\kappa + 1)$ -monitor ($\kappa + 1 \leq |\mathcal{M}|$) placement $\mathcal{O}_{\kappa+1}^*$ can be constructed by $\mathcal{O}_{\kappa+1}^* = \mathcal{O}_{\kappa}^* \cup \{v_m\}$, where $v_m = \arg \max_v N(\mathcal{O}_{\kappa}^* \cup \{v\})$ over $v \in \mathcal{M} \setminus \mathcal{O}_{\kappa}^*$. $\square$

Theorem 6.3 states that for 2-vertex-connected networks, it is always possible to find an optimal κ -monitor placement ($\kappa \geq 3$) within $\mathcal{M}$ (which is a stronger statement than finding an optimal placement within $\mathcal{S}$ as $\mathcal{M}$ is a sub-set of $\mathcal{S}$); moreover, we can always obtain an optimal $(\kappa + 1)$ -monitor placement by expanding an optimal κ -monitor placement with the monitor that maximizes the increase in the number of identifiable links. Applied to GMMP, this result implies that the monitor placement computed by GMMP is optimal as long as $\mathcal{G}$ is 2-vertex-connected (e.g., the monitor placement for the sample 2-vertex-connected network in Fig. 6.5 is optimal, i.e., $\mathcal{O}_3^* = \mathcal{O}_3$, $\mathcal{O}_4^* = \mathcal{O}_4$, $\mathcal{O}_5^* = \mathcal{O}_5$, and $\mathcal{O}_3^* \subset \mathcal{O}_4^* \subset \mathcal{O}_5^*$). We point out that the 2-vertex-connectivity is naturally satisfied by any network that has no single point of failure (no failure of one node will disconnect the network). Nevertheless, we will show in Section 6.3.2 that GMMP is near-optimal even if the network is not 2-vertex-connected.

6.2.4 Complexity of GMMP

The complexity of GMMP depends on the number of monitors κ . If $\kappa = 2$, $\mathcal{O}_2$ is directly obtained by line 2. Since the complexity for computing the number of identifiable links (using Algorithm 5.2) is $O(|V| + |L|)$, and there are $\binom{|\mathcal{S}|}{2}$ possible two-monitor placements, the complexity of line 2 is $O(|\mathcal{S}|^2(|V| + |L|))$. If $\kappa \geq 3$, the computation involves constructing $\mathcal{O}_3$ by line 4 and incrementally expanding it into $\mathcal{O}_{\kappa}$. Line 4 incurs $O(|\mathcal{S}|^3(|V| + |L|))$ complexity for examining identifiability of all the $\binom{|\mathcal{S}|}{3}$ monitor placements. In line 6, the $\arg \max$ operation

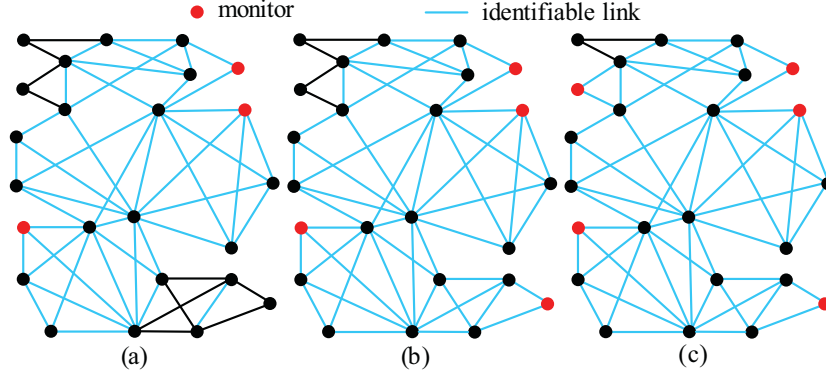


Figure 6.5: Monitor placement by GMMP in a sample network: (a) $\kappa = 3$; (b) $\kappa = 4$; (c) $\kappa = 5$, where $\mathcal{O}_3 \subset \mathcal{O}_4 \subset \mathcal{O}_5$.

involves $|\mathcal{S}| - i + 1$ calls of Algorithm 5.2 to compute v_m , resulting in a complexity of $O(\kappa|\mathcal{S}|(|V| + |L|))$ for lines 5-8. Therefore, the overall complexity for the case of $\kappa \geq 3$ is $O((|\mathcal{S}|^3 + \kappa|\mathcal{S}|)(|V| + |L|))$, which is the same as $O(N_{\mathcal{T}}^3(|V| + |L|))$, as both $|\mathcal{S}|$ and κ ($\kappa < \kappa_{\text{MMP}} \leq |\mathcal{S}|$) are in $O(N_{\mathcal{T}})$ (see Algorithm MMP for the reason of $\kappa_{\text{MMP}} \leq |\mathcal{S}|$). Note that usually $N_{\mathcal{T}} \ll |V|$.

6.2.5 Example

Fig. 6.5 illustrates which nodes are selected as monitors by GMMP in a sample network for $\kappa = 3, 4$, and 5 and shows that the optimal monitor placement follows a nested structure.

6.3 Algorithm Evaluation

6.3.1 Evaluation of MMP

We evaluate MMP through a set of simulations on both randomly-generated and real network topologies. We use the following algorithm, referred to as *Random Monitor Placement (RMP)*, as a benchmark for comparison⁵: Given network $\mathcal{G}$, randomly se-

⁵To our knowledge, MMP is the first monitor placement algorithm for identifying all additive link metrics by measuring controllable, cycle-free paths. Thus, we use random placement to represent the average performance of an arbitrary monitor placement for comparison.

lect κ ($\kappa = 2, \dots, |V|$) nodes as monitors and test the complete identifiability of the resulting network by Theorem 3.7 using the algorithm in Section 3.4.2. Generally, RMP cannot guarantee complete network identifiability for arbitrary $\mathcal{G}$ and κ . Therefore, we measure its performance by the fraction of random placements achieving complete network identifiability over multiple Monte Carlo runs.

6.3.1.1 Random Topologies

We first consider synthetic topologies generated according to four widely used random graph models: Erdős-Rényi (ER) graphs, Random Geometric (RG) graphs, Barabási-Albert (BA) graphs, and Random Power Law (PL) graphs. We randomly generate 100 graph realizations of each model⁶, with each realization containing 150 nodes (i.e., $|V| = 150$). The generated graphs are then input to the monitor placement algorithms. We have introduced ER, RG, and BA graphs in Section 4.6.1 of Chapter 4. Thus, we only explain the PL graph model as stated below.

Random Power Law (PL) graphs: The BA model introduces an artifact that all node degrees are lower bounded by $n_{\min}$. Alternatively, the PL graph [59] provides another way of generating power-law graphs by directly specifying a sequence of expected node degrees $(d_1, \dots, d_{|V|})$ according to the power law, i.e., $d_i = i^\alpha$ ($\alpha > 0$). The generation of a PL graph is similar to that of an ER graph, except that instead of connecting each pair of nodes with the same probability, nodes i and j in a PL graph are connected by a link with probability $p_{ij} = d_i d_j / \sum_{k=1}^{|V|} d_k$.

Remark: Our motivation for performing evaluations on random topologies is that they allow comprehensive evaluation without artifacts of specific network deployments, which are common in real topologies. Moreover, the selected graph models represent diverse networking scenarios (e.g., purely random, wireless, Internet), which provide insights on how the networking scenario affects monitor placement.

⁶All realizations are guaranteed to be connected, as we discard disconnected realizations in the generation process.

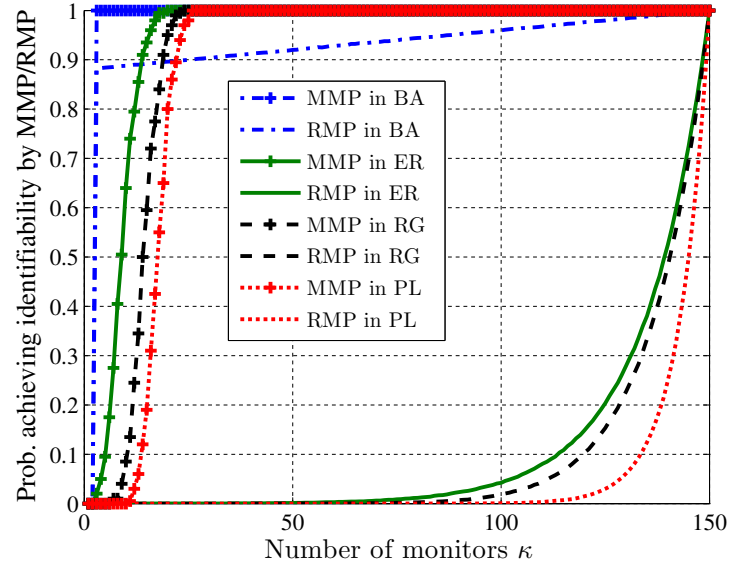


Figure 6.6: Comparison between RMP and MMP: densely-connected graphs ($\bar{n} = 441$, $\bar{\kappa}_{\text{MMP}} = 3$ for BA, $\bar{n} = 437$, $\bar{\kappa}_{\text{MMP}} = 9.36$ for ER, $\bar{n} = 437$, $\bar{\kappa}_{\text{MMP}} = 19.42$ for PL and $\bar{n} = 451$, $\bar{\kappa}_{\text{MMP}} = 14.52$ for RG).

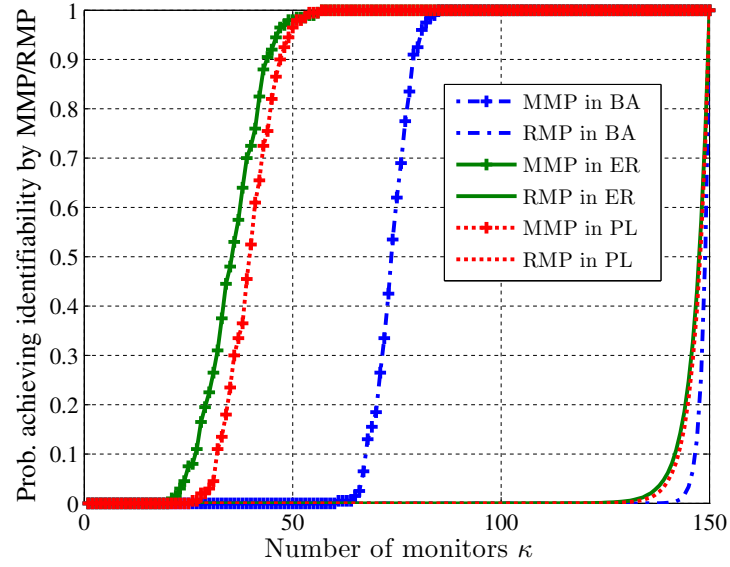


Figure 6.7: Comparison between RMP and MMP: sparsely-connected graphs ($\bar{n} = 295$, $\bar{\kappa}_{\text{MMP}} = 73.51$ for BA, $\bar{n} = 293$, $\bar{\kappa}_{\text{MMP}} = 36.76$ for ER and $\bar{n} = 297$, $\bar{\kappa}_{\text{MMP}} = 40.24$ for PL).

Next, for the four random graph models, we conduct simulations in two scenarios, densely-connected graphs (Fig. 6.6 with parameter configurations: $p = 0.039$ for ER, $d_c = 0.11943$ for RG, $n_{\min} = 3$ for BA, and $\alpha = 0.42$ for PL; see the defi-

nition of parameters p , d_c , and $n_{\min}$ in Section 4.6.1) and sparsely-connected graphs (Fig. 6.7 with parameter configurations: $p = 0.0253$ for ER, $n_{\min} = 2$ for BA, and $\alpha = 0.32$ for PL), to evaluate how graph density affects the performance of MMP and RMP. Since the number of links n and the minimum number of monitors κ_{MMP} (computed by MMP) vary across graph realizations, we present the average values denoted by $\bar{n}$ and $\bar{\kappa}_{\text{MMP}}$ for each graph model, shown in the captions of Fig. 6.6 and 6.7. In each scenario, we have tuned parameters of each model to generate roughly the same average number of links. As RMP is a randomized algorithm, we repeat it for 2000 Monte Carlo runs to obtain the average performance for each graph realization. We then average the results over the 100 graph realizations to obtain the final results⁷, shown in Fig. 6.6 and 6.7.

For densely-connected graphs (Fig. 6.6), the probability that RMP is able to identify all the links increases with the number of monitors κ . However, fewer than 20% of the ER, RG, and PL graphs are identifiable when $\kappa \leq 120$ (recall $|V| = 150$), whereas a careful monitor deployment in the same graphs by MMP ensures identifiability with significantly fewer monitors ($\kappa < 25$). Therefore, in most cases, the proposed algorithm (MMP) substantially outperforms a randomized scheme (RMP) in the required number of monitors. One exception is the BA graphs, where RMP achieves similar performance to that of MMP. This can be explained as follows: When $n_{\min} = 3$ ($n_{\min}$ is the minimum node degree), further simulations show that 87.8% of the generated BA graphs are 3-vertex-connected as each node (except the initial nodes v_2 , v_3 , and v_4) has at least three neighbors; according to MMP, this implies that an arbitrary placement of three monitors will achieve identifiability, and there is no need for a sophisticated placement algorithm. Meanwhile, a comparison of the BA and the PL models implies that the exceptional performance of RMP in the

⁷For each graph realization, MMP achieves identifiability with probability one for $\kappa \geq \kappa_{\text{MMP}}$ and zero for $\kappa < \kappa_{\text{MMP}}$. Therefore, the overall probability for MMP to achieve identifiability using κ monitors is computed as the fraction of graph realizations with $\kappa_{\text{MMP}} \leq \kappa$.

BA model is due to the requirement that all node degrees be lower bounded by three rather than the power law distribution of node degrees, since it does not achieve the same superior performance on PL graphs. For ER, RG, and PL graphs, it is easy to form sparsely-connected sub-graphs, thus requiring more monitors in these sparsely-connected sub-graphs to guarantee network identifiability. Among the four models, we observe that RMP exhibits the worst performance in the model with the largest $\bar{\kappa}_{\text{MMP}}$. This is because large $\bar{\kappa}_{\text{MMP}}$ is resulted from poor connectivity within certain sub-graphs, where a considerable number of nodes have to be selected as monitors to achieve identifiability. Thus, RMP fails to achieve identifiability if it does not select a sufficient number of nodes in any of these sub-graphs.

We perform similar simulations for sparsely-connected graphs by adjusting the parameters⁸, shown in Fig. 6.7. In comparison with Fig. 6.6, both MMP and RMP perform worse in sparsely-connected graphs, requiring a larger number of monitors to achieve the same probability of identifiability. This is because sparser graphs impose more constraints on candidate measurement paths between pairs of monitors, and thus more monitors are needed to identify all links. As we compare the performance of the same graph models in Fig. 6.6 and 6.7, we observe that all models exhibit smooth transitions when we vary the average number of links $\bar{n}$ except for the BA model, which has extremely good performance for $n_{\min} = 3$ but poor performance (except for RG) for $n_{\min} = 2$. This can again be explained by examining the 3-vertex-connectivity of the generated graph realizations. We have verified that as we reduce $n_{\min}$ from 3 to 2, the probability for BA graphs to be 3-vertex-connected drops from 87.8% to 0%, making it unlikely for an arbitrary placement to achieve identifiability. In fact, when $n_{\min} = 2$, on average 49.2% of the nodes in BA graphs have degrees less than 3, which requires them to be selected as monitors by rules (i)

⁸To generate roughly the same number of links as in the other models, d_c for the RG graph becomes much smaller than $\sqrt{\log |V|/(\pi|V|)}$, making the probability of generating a connected RG graph very small. Thus, the RG model is omitted from Fig. 6.7.

and (ii) in MMP. Therefore, RMP fails to achieve identifiability if it misses one of these nodes, resulting in its poor performance.

6.3.1.2 Autonomous System Topologies

We now test MMP and RMP on real network topologies. We use the *Autonomous System* (AS) topologies from both the Rocketfuel [45] (also used in Section 4.6.2) and the CAIDA [46] projects, which represent IP-level connections between backbone/gateway routers of several ASes from major *Internet Service Providers* (ISPs). The parameters of selected networks obtained from these two projects are listed in Tables 6.1 and 6.2, where we sort the networks according to their numbers of links, and $r_{\text{MMP}} := \kappa_{\text{MMP}}/|V|$ denotes the minimum fraction of monitors computed by MMP in a network with $|V|$ nodes.

1) AS topologies from Rocketfuel: As shown in Table 6.1, each AS in Rocketfuel corresponds to an ISP. To identify the entire network, we observe that most ISPs need a significant fraction of nodes to be monitors, ranging from around 30% (Ebone, AT&T, Sprintlink) to more than 60% (Abovenet). This is because ISP networks contain a large number of gateway⁹ routers to connect to customer networks or other ISPs, which appear as dangling nodes that have to be selected as monitors (see rule (i) in Section 6.1). We repeat RMP for $15 \cdot |V|$ Monte Carlo runs for each ISP to evaluate its average performance, measured by the fraction of Monte Carlo runs achieving identifiability, as shown in Fig. 6.8. To facilitate comparison, we mark the fraction of monitors r_{MMP} needed by MMP in the legends of the same plot; note that MMP guarantees identifiability for each network so long as $\kappa/|V| \geq r_{\text{MMP}}$. We normalize the number of monitors by the total number of nodes in order to compare networks of different sizes. Moreover, we only plot the results of RMP for

⁹In real networks, monitor selection may be constrained to a sub-set of nodes, e.g., gateways. Under such constraint, the issue of achievable number of identifiable links (i.e., partial network identifiability) is left for future work.

Table 6.1: Parameters of AS Topologies in Rocketfuel

AS	ISP Name	$ L $	$ V $	κ_{MMP}	r_{MMP}
6461	Abovenet (US)	294	182	117	0.64
1755	Ebone (Europe)	381	172	55	0.32
3257	Tiscali (Europe)	404	240	138	0.58
3967	Exodus (US)	434	201	85	0.42
1221	Telstra (Australia)	758	318	164	0.52
7018	AT&T (US)	2078	631	208	0.33
1239	Sprintlink (US)	2268	604	163	0.27
2914	Verio (US)	2821	960	408	0.43
3356	Level3 (US)	5298	624	94	0.15

$\kappa/|V| \geq 95\%$ since RMP fails to achieve identifiability in almost all of the simulations when $\kappa/|V| < 95\%$.

As in the case of synthetic graphs, we again observe a significant improvement of MMP over RMP. Specifically, RMP has at most 50% probability of identifying all the links even if 99% of the nodes are monitors, whereas MMP guarantees identifiability using at most 64% of nodes as monitors. The poor performance of RMP is due to the heterogeneous connectivity within ISP networks, which contain poorly-connected sub-networks that need a large fraction of monitors, a requirement unlikely to be fulfilled by random placement. We also observe that the relative performance of MMP and RMP varies for different networks, e.g., Level3 and Exodus experience similar performance under RMP, whereas their minimum fractions of monitors computed by MMP differ significantly (0.15 for Level3 and 0.42 for Exodus). Intuitively, this is because RMP performance is determined by the relative number of valid placements, i.e., the fraction of all the $\binom{|V|}{\kappa}$ candidate placements that achieve identifiability, whereas MMP performance is determined by the smallest κ for which this fraction is non-zero.

2) AS topologies from CAIDA: ISP topologies have evolved since the Rocketfuel project; consequently, we repeat the above evaluation on a recent data set obtained

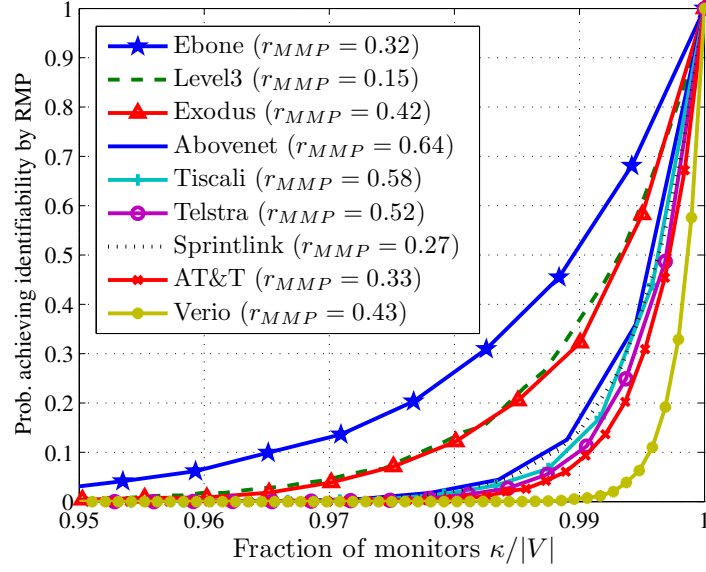


Figure 6.8: Comparison between RMP and MMP: ISP topologies in Rocketfuel ($15 \cdot |V|$ Monte Carlo runs, $r_{MMP} := \kappa_{MMP}/|V|$).

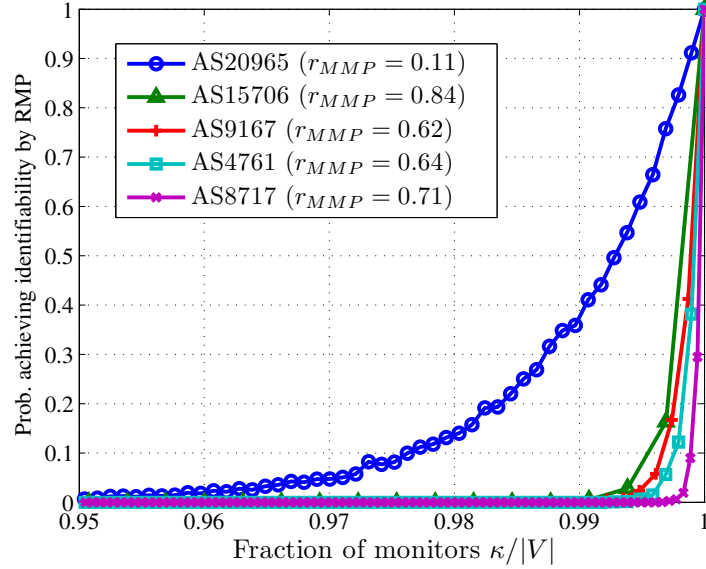


Figure 6.9: Comparison between RMP and MMP: AS topologies in CAIDA (3000 Monte Carlo runs, $r_{MMP} := \kappa_{MMP}/|V|$).

by the CAIDA project; see results in Table 6.2 and Fig. 6.9. Compared with the ASes in Rocketfuel data set, we observe that ASes with similar average node degrees (i.e., $2|L|/|V|$) in the CAIDA data set require a larger fraction of monitors for complete identification, e.g., $r_{MMP} = 0.71$ for AS8717 with average node degree 4.2 in

Table 6.2, whereas $r_{\text{MMP}} = 0.32$ for AS1755 (Ebone) with average node degree 4.4 in Table 6.1. This is because ASes in the CAIDA data set tend to be more skewed in connectivity, with more densely-connected cores and a larger number of dangling nodes (likely the gateways for peer/customer connections). As all dangling nodes have to be selected as monitors, such skewed connectivity leads to a higher ratio of monitors although the average node degree is not necessarily smaller. Meanwhile, comparison with RMP (3000 Monte Carlo runs), presented in Fig. 6.9, shows that the improvement of MMP over RMP remains significant. As in Fig. 6.8, RMP again exhibits poor performance for all the ASes considered in Fig. 6.9, which has less than 35% probability of identifying all links even if the fraction of monitors $\kappa/|V|$ is as large as 0.99. In particular, for almost all the networks (except for AS20965) in Fig. 6.9, RMP fails to identify all the links in more than 60% of the simulations even if *all but one* node are monitors. In contrast, carefully selected monitors by MMP can guarantee complete identification while substantially reducing the required number of monitors.

We note that AS topologies from both Rocketfuel and CAIDA contain a large fraction of dangling nodes, which have to be selected as monitors to ensure network complete identifiability. Simulation results show if we remove all these dangling nodes and only interested in identifying the link metrics in the remaining graph (e.g., the backbone network), then the required number of monitors can be substantially reduced. This implies that the network administrators can tailor the network topology according to the practical monitoring requirement (e.g., monitoring the links with high importance), and then feed this well-tailored topology to MMP for computing the optimal monitor placement.

Table 6.2: Parameters of AS Topologies in CAIDA

AS	$ L $	$ V $	κ_{MMP}	r_{MMP}
15706	874	325	276	0.84
9167	1590	769	483	0.62
8717	3755	1778	1266	0.71
4761	3760	969	624	0.64
20965	8283	968	110	0.11

6.3.2 Evaluation of GMMP

We evaluate GMMP also on the *Internet Service Provider (ISP)* topologies collected by the Rocketfuel project [45]. We select three ISPs of similar sizes to evaluate the impact of different topologies; see Table 6.3 for the details, where N_B denotes the number of biconnected components, and N_T the number of triconnected components. We apply GMMP to each ISP, with a focus on evaluating the *identification ratio* (ratio of number of identifiable links over total number of links) to compare networks of different sizes. We also evaluate algorithm efficiency, measured by its running time.

As a benchmark for GMMP, we employ a similar algorithm used for evaluating MMP (i.e., RMP), referred to as *Advanced Random Monitor Placement (ARMP)*: Given a total number of monitors κ and a node set Φ , ARMP randomly selects κ nodes out of Φ as monitors and examines the corresponding number of identifiable links by DAIL (Algorithm 5.2). This process is repeated δ times, and the selection yielding the largest number of identifiable links is returned as the final placement by ARMP. The basic version for ARMP, denoted by ARMP- V , is to select nodes from the entire network, i.e., $\Phi = V$. A more advanced version of ARMP, denoted by ARMP- S , limits selection to the candidate set S , i.e., $\Phi = S$, based on the observation in Theorem 6.2. We employ both versions of ARMP to evaluate the performance impact of using a refined candidate set. As a randomized alternative to exhaustive

Table 6.3: ISP Networks for Evaluation of GMMP

ISP	$ L $	$ V $	$ \mathcal{S} $	$N_{\mathcal{B}}$	$N_{\mathcal{T}}$	κ_{MMP}	r_{MMP}
Abovenet (US)	294	182	148	87	129	117	0.64
EBONE (Europe)	381	172	81	28	65	55	0.32
Exodus (US)	434	201	108	38	92	85	0.42

enumeration, ARMP trades off complexity against performance via tuning parameter δ . In this evaluation, we set δ to $\kappa|\mathcal{S}|$ for both versions of ARMP, which implies a complexity of $O(\kappa|\mathcal{S}|(|V| + |L|))$. We notice that GMMP in its current form (Algorithm 6.3) will incur a higher complexity due to the enumeration of all 3-monitor placements (line 4); in fact, even a complete enumeration of 2-monitor placements (line 2) will incur complexity of $O(|\mathcal{S}|^2(|V| + |L|))$, which is already higher than the complexity of ARMP for small κ . To ensure fair comparison with ARMP, we simplify GMMP by only examining $\kappa|\mathcal{S}|$ randomly selected 2-monitor placements and using the one with the maximum number of identifiable links to approximate $\mathcal{O}_2$; $\mathcal{O}_3, \dots, \mathcal{O}_\kappa$ are then incrementally constructed based on $\mathcal{O}_2$ by repeating lines 6–7 of Algorithm 6.3. In the following, all results labeled “GMMP” are for this simplified GMMP. We perform the simulations using Matlab R2010a on a computer with Intel(R) Core i7-2720QM CPU @ 2.2GHz, 16.0 GB memory, and 64-bit Win7 OS.

We report the identification ratio in Fig. 6.10, where the optimal values are obtained by an exhaustive search over all κ -monitor placements¹⁰. We make the following observations:

1) Although complete identification of the ISP networks requires a substantial fraction of nodes to be monitors (0.32 to 0.64 as shown in Table 6.3), many links can be identified with only a few monitors, e.g., both GMMP and ARMP- $\mathcal{S}$ identify more than half of the links in all the ISPs by placing monitors at only 5% of the nodes. This is because most ISP networks contain at least one large sub-network

¹⁰In Fig. 6.10, we only simulate $\kappa/|V| \in (0, \kappa_{\text{MMP}}/|V|)$, as the identification ratio of 1 can always be achieved by MMP for $\kappa/|V| \geq \kappa_{\text{MMP}}/|V|$.

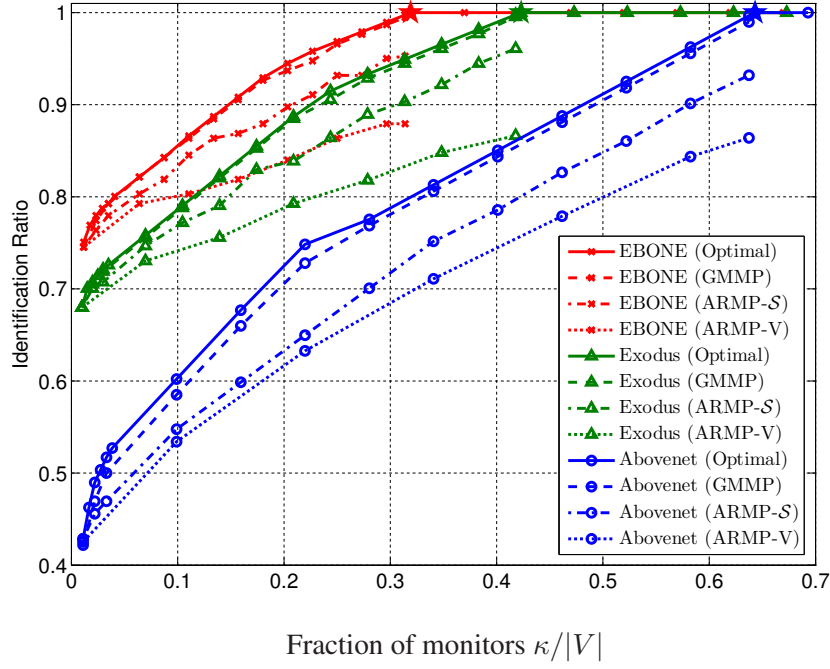


Figure 6.10: Identification ratio in ISP networks; $\star : (\kappa_{\text{MMP}}/|V|, 1)$, i.e., transition point for complete identification.

(likely the backbone) that tends to be 3-vertex-connected, which can be completely identified using a small number of properly placed monitors.

2) GMMP can be suboptimal for non-2-vertex-connected networks (none of the ISPs is 2-vertex-connected, as verified by $N_B > 1$ in Table 6.3). Nevertheless, it closely approximates the optimal identification ratio for all the networks over all considered values of κ . This observation suggests that, besides being provably optimal for 2-vertex-connected networks, GMMP is also near-optimal for general networks (note that this is by the simplified GMMP, and the identification ratio of the original GMMP can be higher). In comparison, ARMP-V shows a notably lower identification ratio, e.g., to reach identification ratio 0.85 in EBONE, 16 monitors are required by GMMP, while 39 are required by ARMP-V. Notice that ARMP-S achieves intermediate performance, e.g., requiring 21 monitors in EBONE to achieve identification ratio 0.85. For the same set of evaluations, we have also simulated al-

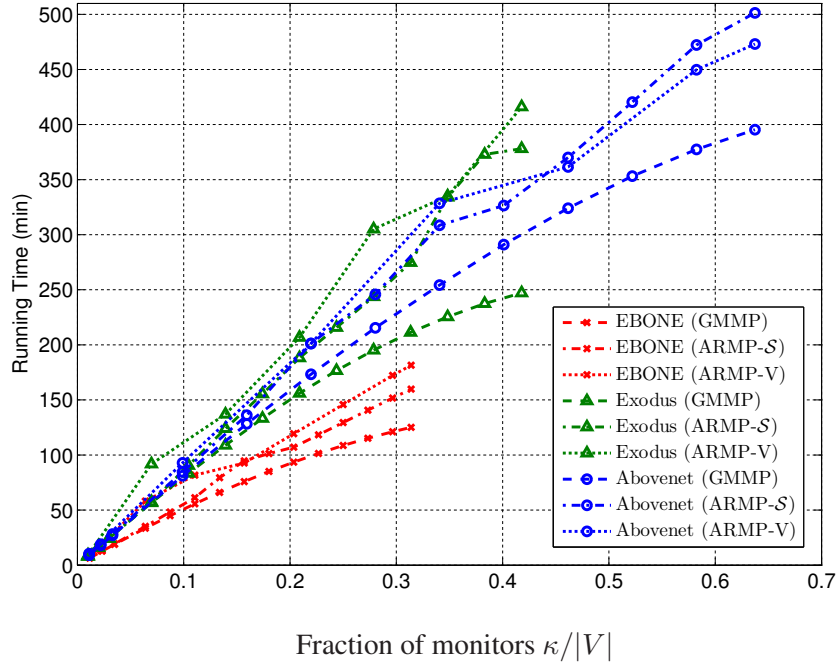


Figure 6.11: Algorithm running time in ISP networks.

gorithm running time in Fig. 6.11 which shows¹¹ that GMMP is faster than ARMP-V and ARMP-S.

These results not only confirm the performance improvement of GMMP over the benchmarks, but also reveal the factors contributing to this improvement: refined set of candidate monitors (shown by the improvement of ARMP-S over ARMP-V) and strategic selection of monitors from the candidate set (shown by the improvement of GMMP over ARMP-S). Note that all the algorithms, including the exhaustive search, depend on the proposed algorithm DAIL to evaluate the number of identifiable links under each candidate placement, which otherwise cannot be determined efficiently (see discussions at the beginning of Chapter 5).

3) Among the three ISPs, we see that EBONE has the highest identification ratio while Abovenet has the lowest. Intuitively, this is because EBONE is more densely connected (as shown by a smaller N_B in Table 6.3), which gives more flexibility in

¹¹In Fig. 6.11, the running time for an exhaustive search is significantly larger (over 48 hours for each κ), and is thus omitted in Fig. 6.11.

constructing measurement paths between monitors, and thus requires fewer monitors for identifying its links; in contrast, Abovenet is poorly connected with a large number of cut-vertices (as shown by a larger N_B), which limits the set of measurable paths between each pair of monitors and thus requires more monitors. This observation suggests that networks with better connectivity are more amenable to network tomography under limited monitor resources.

Conclusions and Future Work

7.1 Conclusions

Network tomography is an effective approach that determines network internal information through external end-to-end path measurements. Though providing a lightweight solution in revealing network internal states without direct performance measurement, network tomography still involves several key issues that are not well understood. Therefore, in this thesis, we studied a set of fundamental problems that are essential to network tomography, with particular interest in additive link metric identification from both theoretical and algorithmic perspectives. To comply with practical routing restrictions, each research issue in this thesis was explored under the constraint that all measurement paths be cycle-free (simple paths). Furthermore, we emphasized *efficiency* for each research issue. Specifically, we characterized network identifiability by easily verifiable topological conditions and developed highly efficient algorithms for path construction, link identification, and network (partial) identifiability determination/maximization, thus enabling applicable solutions in building robust monitoring infrastructures.

In this thesis, to the best of our knowledge, we established the first fundamental conditions on network topology and placement of monitors for identifying additive link metrics using end-to-end measurements along simple paths between monitors. We showed that with two monitors, it is impossible to identify all the link metrics; however, it is possible to identify the metrics of interior links that are at least

one hop away from the monitors, for which we derived the necessary and sufficient conditions in terms of edge/vertex connectivity of the network topology. Next, we studied the case of three or more monitors and derived the corresponding necessary and sufficient conditions for identifying all the link metrics. We showed that these conditions are not only useful for testing network identifiability under a given monitor placement, but also enable efficient algorithm design for link identification and monitor placement.

To obtain all link metrics in an identifiable network, we then proposed efficient algorithms for constructing measurement paths and uniquely identifying link metrics from path measurements. The proposed algorithms utilize a special structure of identifiable networks in the form of independent spanning trees to strategically construct linearly independent measurement paths and compute link metrics without explicitly inverting the measurement matrix. Extensive simulations on both synthetic and real networks show that the proposed algorithms can guarantee unique identification of link metrics while being orders of magnitude faster than existing solutions for large networks.

Considering the fact that some links remain identifiable by measuring cycle-free paths even if the entire network is not completely identifiable, we further investigated network partial identifiability by developing linear-time algorithm (called DAIL) that locates all identifiable links under given monitor placement. We later showed that this algorithm serves as an efficient sub-routine in designing the algorithm for optimum monitor placement.

Finally, we studied the problem of maximal identifiability monitor placement in network tomography. To guarantee complete identifiability, we proposed a monitor placement algorithm using the minimum number of monitors. In cases where monitor budget is insufficient, we developed an efficient greedy algorithm that incrementally places monitors to maximize the number of identifiable links based on the

linear-time algorithm DAIL. Our evaluations on real network topologies verified that the proposed monitor placement algorithms significantly outperform baseline solutions. Moreover, the proposed partial monitor placement algorithm is both provably optimal for networks without cut-vertices and empirically near-optimal for several real network topologies, thus providing a quantifiable tradeoff between the level of identifiability and the number of monitors.

7.2 Future Work

With respect to more practical network conditions, this research work can be further extended in the future. Considering that network elements may fail to provide normal communication functionalities during network operations, it is important to localize and correct these network failures in a timely manner. To localize potential failures (non-additive metrics), most existing work experiences limitations in applicability. For instance, algorithm Smallest Consistent Failure Set (SCFS) [60, 61] is only applicable to tree topologies and heuristic solutions in [62] exhibit severe accuracy issue. Hence, there is still a lack of understanding in characterizing network failure localizability by network topologies and monitor placements. This issue is substantially different from the link metric identification in this thesis in that [63] shows generally more measurement paths are required for localizing link failures. Furthermore, when the network topology and the network states are unstable, the accuracy of network state inference can be deteriorated. Thus, a tomographic solution which is capable of adapting to dynamic environments is also a necessity.

All these requirements in network tomography, therefore, motivate future research for solving the following key research issues.

1) Link identification under more constraints: We have identified topological conditions for uniquely identifying link metrics from path measurements and devel-

oped an efficient algorithm (MMP) that places the minimum number of monitors to satisfy the identifiability condition. These results, however, assume that the network is modeled as a stable undirected graph and all measurement paths are arbitrarily controllable. In cases that some nodes/links may fail, link metrics are asymmetric, or measurable paths are (partially) uncontrollable, a follow-up question is: Under these challenging network settings, which internal (link/node) metrics are identifiable and where should the monitors be placed?

2) Determination of network failure localizability: For this issue, two closely-related questions can be addressed. (i) Given arbitrary networks, under what conditions can we uniquely localize failed nodes? (ii) How many failed nodes can be localized via end-to-end cycle-free path measurements?

3) Dynamic model for characterizing topology changes in mobile networks: In hybrid communication networks, network topologies may vary over time. To apply network tomography in dynamic environments, an efficient method is required to predict network topologies in the following time slots based on observations/samplings. Accordingly, network administrators can then optimize the measurement path selection and monitor deployment, aiming to maximize network identifiability.

Bibliography

- [1] F. Lo Presti, N. Duffield, J. Horowitz, and D. Towsley, “Multicast-based inference of network-internal delay distributions,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 761–775, Dec. 2002. [1](#), [7](#)
- [2] A. B. Downey, “Using pathchar to estimate internet link characteristics,” in *ACM SIGCOMM*, 1999. [2](#), [8](#)
- [3] G. Jin, G. Yang, B. Crowley, and D. Agarwal, “Network characterization service (NCS),” in *IEEE HPDC*, 2001. [2](#)
- [4] Y. Vardi, “Estimating source-destination traffic intensities from link data,” *Journal of the American Statistical Assoc.*, pp. 365–377, 1996. [2](#)
- [5] M. Coates, A. O. Hero, R. Nowak, and B. Yu, “Internet tomography,” *IEEE Signal Processing Magazine*, vol. 19, pp. 47–65, 2002. [2](#)
- [6] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, “Traffic matrix estimation: Existing techniques and new directions,” in *ACM SIGCOMM*, 2002. [2](#)
- [7] E. Lawrence and G. Michailidis, “Network tomography: A review and recent developments,” *Frontiers in Statistics*, vol. 54, 2006. [2](#), [7](#), [9](#), [53](#)
- [8] O. Gurewitz and M. Sidi, “Estimating one-way delays from cyclic-path delay measurements,” in *IEEE INFOCOM*, 2001. [3](#), [7](#), [8](#)
- [9] Y. Chen, D. Bindel, H. Song, and R. H. Katz, “An algebraic approach to practical and scalable overlay network monitoring,” in *ACM SIGCOMM*, 2004. [3](#), [7](#), [8](#), [61](#)

- [10] A. Chen, J. Cao, and T. Bu, “Network Tomography: Identifiability and Fourier domain estimation,” in *IEEE INFOCOM*, 2007. 3, 8
- [11] L. Ma, T. He, K. Leung, D. Towsley, and A. Swami, “Efficient identification of additive link metrics via network tomography,” in *ICDCS*, 2013. 3
- [12] G. H. Golub and C. F. Van-Loan, *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 1996. 5, 9, 46
- [13] N. Duffield and F. Lo Presti, “Multicast inference of packet delay variance at interior network links,” in *IEEE INFOCOM*, 2000. 7, 9
- [14] Y. Xia and D. Tse, “Inference of link delay in communication networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2235–2248, 2006. 7, 8
- [15] A. Adams, T. Bu, T. Friedman, J. Horowitz, D. Towsley, R. Caceres, N. Duffield, F. Presti, and V. Paxson, “The use of end-to-end multicast measurements for characterizing internal network behavior,” *IEEE Communications Magazine*, vol. 38, no. 5, pp. 152–159, May 2000. 7, 9
- [16] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, “Network tomography: recent developments,” *Statistical Science*, 2004. 7, 9
- [17] J. Ni and S. Tatikonda, “Network tomography based on additive metrics,” *IEEE Transactions on Information Theory*, vol. 57, pp. 7798–7809, 2011. 7
- [18] M.-F. Shih and A. Hero, “Unicast inference of network link delay distributions from edge measurements,” in *IEEE ICASSP*, 2001. 7
- [19] T. Bu, N. Duffield, and F. Lo Presti, “Network tomography on general topologies,” in *ACM SIGMETRICS*, 2002. 7

- [20] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, “Identifiability of link metrics based on end-to-end path measurements,” in *ACM IMC*, 2013. 8
- [21] S. Ahuja, S. Ramasubramanian, and M. Krunz, “SRLG failure localization in all-optical networks using monitoring cycles and paths,” in *IEEE INFOCOM*, 2008. 8
- [22] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, “Locating Internet bottlenecks: Algorithms, measurements, and implications,” in *ACM SIGCOMM*, 2004. 8
- [23] J. D. Horton and A. López-Ortiz, “On the number of distributed measurement points for network tomography,” in *ACM IMC*, 2003. 8
- [24] R. Kumar and J. Kaur, “Practical beacon placement for link monitoring using network tomography,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2196–2209, Dec. 2006. 8
- [25] Y. Bejerano and R. Rastogi, “Robust monitoring of link delays and faults in IP networks,” in *IEEE INFOCOM*, 2003. 8, 10, 79
- [26] 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. 8, 9, 21
- [27] N. Alon, Y. Emek, M. Feldman, and M. Tennenholtz, “Economical graph discovery,” in *Symposium on Innovations in Computer Science*, 2011. 8
- [28] M. Firooz and S. Roy, “Network tomography via compressed sensing,” in *IEEE GLOBECOM*, 2010. 9
- [29] W. Xu, E. Mallada, and A. Tang, “Compressive sensing over graphs,” in *IEEE INFOCOM*, 2011. 9

- [30] A. Gopalan and S. Ramasubramanian, “On the maximum number of linearly independent cycles and paths in a network,” Technical Report, August 2012. [Online]. Available: <http://www2.engr.arizona.edu/%7esrini/papers/lirank.pdf> 9
- [31] O. Wing and W. Kim, “The path matrix and its realizability,” *IRE Transactions on Circuit Theory*, vol. 6, pp. 267–272, 1959. 10
- [32] N. Jacobson, *Basic Algebra I: Second Edition*. Dover Publications, 2009. 10
- [33] Y. Zhao, Y. Chen, and D. Bindel, “Towards unbiased end-to-end network diagnosis,” in *ACM SIGCOMM*, 2006. 10
- [34] Y. Shavitt, X. Sun, A. Wool, and B. Yener, “Computing the unmeasured: an algebraic approach to internet mapping,” *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 67–78, 2004. 10
- [35] R. Mahajan, N. T. Spring, D. Wetherall, and T. E. Anderson, “Inferring link weights using end-to-end measurements,” in *ACM Internet Measurement Workshop*, 2002. 10
- [36] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, “Link identifiability in communication networks with two monitors,” in *IEEE GLOBECOM*, 2013. 10
- [37] R. Kumar and J. Kaur, “Practical beacon placement for link monitoring using network tomography,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2196–2209, 2006. 10, 79
- [38] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, “Monitor placement for maximal identifiability in network tomography,” in *IEEE INFOCOM*, 2014. 10
- [39] H. X. Nguyen and P. Thiran, “Active measurement for multiple link failures diagnosis in IP networks,” in *Passive and Active Measurement*, 2004. 10

- [40] J. D. Horton and A. Lopez-Ortiz, “On the number of distributed measurement points for network tomography,” in *ACM IMC*, 2003. 10, 79
- [41] [Online]. Available: <http://www.ietf.org/rfc/rfc0791.txt> 11
- [42] “Open networking foundation.” [Online]. Available: <http://www.opennetworkingfoundation.org> 11
- [43] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, “B4: Experience with a globally-deployed Software Defined WAN,” in *ACM SIGCOMM*, 2013. 11
- [44] D. Mcdysan, “Software Defined Networking opportunities for transport,” *IEEE Communications Magazine*, vol. 51, pp. 28–31, 2013. 11
- [45] “Rocketfuel: An ISP topology mapping engine,” University of Washington, 2002. [Online]. Available: <http://www.cs.washington.edu/research/networking/rocketfuel/> 12, 58, 95, 99
- [46] “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/> 12, 95
- [47] R. Diestel, *Graph theory*. Springer-Verlag Heidelberg, New York, 2005. 12, 23, 36, 41, 142
- [48] H. Nguyen and P. Thiran, “The boolean solution to the congested IP link location problem: Theory and practice,” in *IEEE INFOCOM*, 2007. 14
- [49] H. X. Nguyen and P. Thiran, “Network loss inference with second order statistics of end-to-end flows,” in *ACM IMC*, 2007. 14

- [50] R. E. Tarjan, “A note on finding the bridges of a graph,” *Information Processing Letters*, pp. 160–161, 1974. 37
- [51] J. E. Hopcroft and R. E. Tarjan, “Dividing a graph into triconnected components,” *SIAM Journal on Computing*, vol. 2, pp. 135–158, 1973. 37, 38, 64, 65, 69, 73, 78, 79, 80, 122, 139, 170
- [52] J. Cheriyan and S. N. Maheshwari, “Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs,” *Journal of Algorithms*, vol. 9, pp. 507–537, 1988. 42, 43, 48, 51, 119, 120
- [53] J. H. Wilkinson, “Error analysis of direct methods of matrix inversion,” *Journal of the ACM*, vol. 8, pp. 281–330, 1961. 46
- [54] D. Gleich, “MatlabBGL: A Matlab graph library,” October 2008. [Online]. Available: https://www.cs.purdue.edu/homes/dgleich/packages/matlab_bgl/ 54
- [55] P. Erdős 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. 54
- [56] P. Gupta and P. Kumar, “Critical power for asymptotic connectivity in wireless networks,” *Stochastic Analysis, Control, Optimization and Applications*, pp. 547–566, 1999. 55
- [57] R. Albert and A.-L. Barabási, “Statistical mechanics of complex networks,” *Reviews of Modern Physics*, vol. 74, pp. 47–97, Jan. 2002. 55
- [58] R. Tarjan, “Depth-first search and linear graph algorithms,” *SIAM Journal on Computing*, vol. 1, pp. 146–160, 1972. 64, 65, 69, 73, 77, 79, 122
- [59] F. Chung and L. Lu, *Complex Graphs and Networks*. American Mathematical Society, 2006. 91

-
- [60] N. Duffield, “Simple network performance tomography,” in *ACM IMC*, 2003. 106
- [61] —, “Network tomography of binary network performance characteristics,” *IEEE Transactions on Information Theory*, vol. 52, pp. 5373–5388, 2006. 106
- [62] 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. 106
- [63] H. X. Nguyen and P. Thiran, “Binary versus analogue path monitoring in IP networks,” in *Passive and Active Measurement*, 2005. 106
- [64] S. Even and R. E. Tarjan, “Computing an st-numbering,” *Theoretical Computer Science*, vol. 2, pp. 339–344, 1976. 119, 120

Auxiliary Algorithms

Auxiliary algorithms for DAIL (see Algorithm 5.2 in Chapter 5) are presented in this appendix. To facilitate the presentation, we first introduce a notion to distinguish between two types of vantages as follows:

Definition 14. A vantage v is *conjugate* with another vantage v' (called a *conjugate pair*) if (i) $\{v, v'\}$ forms a Type-1-VC, and (ii) neither of them is an agent.

The significance of this classification of vantages is that for a given triconnected component, there exists a set of (external) paths (see Lemma D.4 in Appendix D) connecting each vantage to an agent such that the path for each independent vantage (see Definition 13 in Chapter 6) is vertex disjoint with all the other paths, whereas the paths for vantages in a conjugate pair may lead to the same agent but are disjoint elsewhere (e.g., $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_3\}$ in Fig. A.1 (c) are conjugate pairs). Note that the definition of conjugate pairs only implies that vantages in a conjugate pair must connect to the same agent if their paths are constrained to the same neighboring sub-graph of the parent biconnected component behind the cut (from the perspective of the triconnected component under consideration). Without this constraint, it is still possible to find disjoint paths for vantages in conjugate pairs, e.g., conjugate vantages μ_1 and μ_2 in Fig. A.1 (c) can connect to agents via disjoint paths $\mathcal{P}_4$ and $\mathcal{P}_2$, respectively.

Based on the type of vantages, we further classify triconnected component of Category 2 into 4 sub-categories:

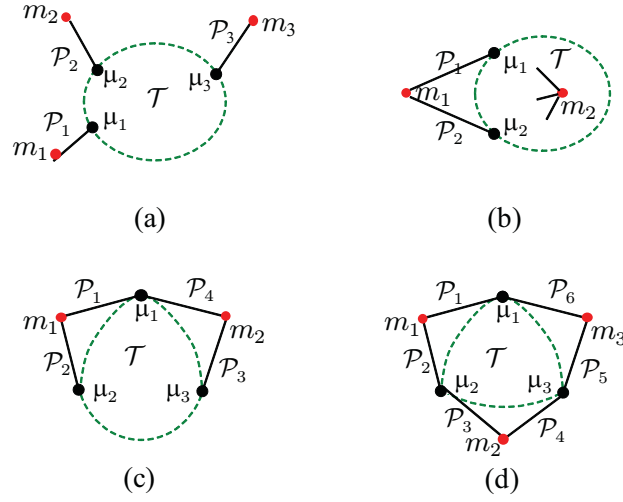


Figure A.1: Four sub-categories of Category 2.

- (i) **Category 2.1:** containing only 3 vantages, at least two of which are independent (see Definition 13 in Chapter 6), e.g., $\mathcal{T}$ in Fig. A.1 (a);
- (ii) **Category 2.2:** containing only 3 vantages, in which only one is independent and the other two form a conjugate pair, e.g., $\mathcal{T}$ in Fig. A.1 (b);
- (iii) **Category 2.3:** containing only 3 vantages $\{\mu_1, \mu_2, \mu_3\}$, in which none of them is independent, and only $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_3\}$ are conjugate pairs, e.g., $\mathcal{T}$ in Fig. A.1 (c);
- (iv) **Category 2.4:** containing only 3 vantages, in which none of them is independent and each vantage pair forms a conjugate pair, e.g., $\mathcal{T}$ in Fig. A.1 (d);

The strategy of DAIL is to apply Theorem 3.3 or 3.7 to each triconnected component together with its associated vantage-to-agent connections. Moreover, the prerequisite of applying Theorem 3.3 and 3.7 to the (sub-)graph of interest is that all involved links can be employed for constructing measurement paths. However, for an individual triconnected component, it may involve virtual links, which do not exist in the original graph $\mathcal{G}$. To handle this issue, we prove in Appendix E that for Categories 1, 2.2, 2.4 and 3, even if the associated triconnected component contains virtual links, the conclusions on the sub-graph identifiability in DAIL still hold. However, triangle components of Category 2.1 and 2.3 are special cases requiring

Algorithm A.1: Determination of All Identifiable Links in Triangles of Category 2.1

input : Triangle component $\mathcal{T}$ of Category 2.1, and the neighboring triconnected components of $\mathcal{T}$

output: Identifiable links in $\mathcal{T}$

```

1 foreach real link  $l_i$  in  $\mathcal{T}$  do
2   Suppose  $l_i$  is incident to  $v_1$  and  $v_2$  and the third node in  $\mathcal{T}$  is  $v_3$ . Let  $S_1$  ( $S_2$ ) be
   the set of immediate neighboring triconnected components connected to  $\mathcal{T}$  via
    $\{v_1, v_3\}$  ( $\{v_2, v_3\}$ );
3   Let  $\mathcal{T}_1^*$  ( $\mathcal{T}_2^*$ ) be a neighboring triconnected component of  $\mathcal{T}$  with the following
   properties: (i)  $v_1 \in \mathcal{T}_1^*$  ( $v_2 \in \mathcal{T}_2^*$ ); (ii)  $v_2 \notin \mathcal{T}_1^*$  ( $v_1 \notin \mathcal{T}_2^*$ ); (iii)  $\mathcal{T}_1^*$  ( $\mathcal{T}_2^*$ ) contains
   at least two real links which are incident to  $v_1$  ( $v_2$ ); (iv)  $\mathcal{T}_1^*$  ( $\mathcal{T}_2^*$ ) contains 3 or
   more vantages;
4   foreach  $j$  ( $j = 1, 2$ ) do
5     if  $v_j v_3$  is virtual AND  $v_j$  is not an agent in the parent biconnected component
     of  $\mathcal{T}$  AND  $\{v_1, v_2\}$  is a Type-0-VC w.r.t.  $\mathcal{T}$  if  $\{v_1, v_2\}$  is a 2-vertex cut AND
      $|S_j| = 1$  AND  $\mathcal{T}_j^*$  satisfying the properties in line 3 does not exist then
6        $l_i$  is unidentifiable; break;
7     end
8   end
9   if  $l_i$  is not marked as unidentifiable by line 6 then
10     $l_i$  is identifiable;
11  end
12 end

```

separate treatment. Therefore, we introduce auxiliary algorithms, Algorithm A.1 and A.2. The key question that Algorithm A.1 and A.2 answers is: when virtual links are used in identifying a link, are there real paths in neighboring components to replace these virtual links? Consider a triangle of Category 2.1 (or 2.3) with vantages μ_1 , μ_2 and μ_3 . To identify link $\mu_2\mu_3$, the path replacement of $\mu_1\mu_2$ or $\mu_1\mu_3$ (if they are virtual) may involve agents, which implies that the condition in Theorem 3.7 (or Condition ② in Theorem 3.3) is not satisfied (see the proof in Appendix E). Therefore, we build the conditions in line 9 of Algorithm A.1 (and line 2 of Algorithm A.2) to ensure the existence of path replacements that do not contain any agents. Specifically, if a triconnected component of Category 2.1 satisfies the conditions in line 5 (Algorithm A.1), then there is no path replacement of $\mu_1\mu_2$ or $\mu_1\mu_3$, and thus we can at most identify the sum metric of a path and link $\mu_2\mu_3$, but

Algorithm A.2: Determination of All Identifiable Links in Triangles of Category 2.3

input : Triangle component $\mathcal{T}$ of Category 2.3 with two conjugate pairs $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_3\}$, and a set S_1 (S_2) of its immediately neighboring triconnected components connected to $\mathcal{T}$ via $\{\mu_1, \mu_2\}$ ($\{\mu_1, \mu_3\}$)

output: Identifiable links in $\mathcal{T}$

- 1 $\mu_1\mu_2$ ($\mu_1\mu_3$) is identifiable if it is a real link;
- 2 **if** (*link $\mu_1\mu_2$ is real OR $|S_1| \geq 2$ OR one component in S_1 is 3-vertex-connected*)
AND (*link $\mu_1\mu_3$ is real OR $|S_2| \geq 2$ OR one component in S_2 is 3-vertex-connected*)
then
- 3 | $\mu_2\mu_3$ is identifiable if it is a real link;
- 4 **end**

Algorithm A.3: Determination of All Identifiable Links in Triangles of Category 2

input : Triangle component $\mathcal{T}$ of Category 2 and its neighboring triconnected components

output: Identifiable links in $\mathcal{T}$

- 1 **if** $\mathcal{T}$ is of Category 2.1 **then**
- 2 | the identifiability of $\mathcal{T}$ is determined by Algorithm A.1;
- 3 **else if** $\mathcal{T}$ is of Category 2.2 **then**
- 4 | all links except for the ones incident to the independent vantage in $\mathcal{T}$ are identifiable;
- 5 **else if** $\mathcal{T}$ is of Category 2.3 **then**
- 6 | the identifiability of $\mathcal{T}$ is determined by Algorithm A.2;
- 7 **else** // $\mathcal{T}$ must be of Category 2.4
- 8 | all links in $\mathcal{T}$ are identifiable;
- 9 **end**

not individual metrics. For a triangle of Category 2.3 with conjugate pairs $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_3\}$, nevertheless, any path replacement of $\mu_2\mu_3$ in neighboring components of $\mathcal{T}$ does not involve agents. Thus, this freedom of path selections benefits the identification of $\mu_1\mu_2$ and $\mu_1\mu_3$. In fact, $\mu_1\mu_2$ and $\mu_1\mu_3$ (if any) are always identifiable (line 1 of Algorithm A.2). The detailed proof of Algorithms A.1 and A.2 are presented in Appendix D (Lemmas D.5 and D.6). Since the four sub-categories of Category 2 can cover all Category 2 cases, we have Algorithm A.3 to identify any triangle component of Category 2.

Independent Spanning Trees

In this appendix, we briefly discuss how to construct the 3 independent spanning trees in an 3-vertex-connected r -extended graph $\mathcal{G}_{ex}^*$, some unique features of which will be used in the proofs of Lemma D.3 (Appendix D) and Theorem 4.2.

Definition 15. ([52]) *Non-separating ear decomposition:* Non-separating ear decomposition of $\mathcal{G}$ is a decomposition $V(\mathcal{G}) = V(\mathcal{E}_1 \cup \mathcal{E}_2 \cup \dots \cup \mathcal{E}_{n_e})$ (each $\mathcal{E}_i$ is called an *ear*, n_e ears in total) such that

1. $\mathcal{E}_1$ is an induced cycle,
2. $\mathcal{E}_i$ ($2 \leq i \leq n_e$) is an induced simple path with only its end-points in common with $\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}$,
3. Let $\overline{\mathcal{G}}_i := \mathcal{G} \setminus (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \dots \cup \mathcal{E}_i)$. For each $\mathcal{E}_i$, $\overline{\mathcal{G}}_i$ is connected and each internal node of $\mathcal{E}_i$ has a neighbor in $\overline{\mathcal{G}}_i$, and
4. $|\mathcal{E}_{n_e}| = 3$ and the internal node of $\mathcal{E}_{n_e}$ does not appear in other ears.

Definition 16. ([64]) *s-t numbering:* s-t numbering of $\mathcal{G}$ is a one-one function $f : V \rightarrow [1 \dots k]$ ($k = |V|$) such that

1. $f(s) = 1$, $f(t) = k$, and
2. For each vertex v in $\mathcal{G} \setminus \{s, t\}$, it has a neighbor u with $f(u) < f(v)$ and a neighbor w with $f(w) > f(v)$.

The non-separating ear decomposition and s-t numbering are fundamental in constructing independent spanning trees. Suppose $\mathcal{E}_j$ is the first ear with node v on it, then j is called the *ear level* of v , denoted by $g(v) = j$. It is proved in [52] that a

3-vertex-connected graph $\mathcal{G}_{ex}^*$ has a nonseparation ear decomposition, by which each node in $\mathcal{G}_{ex}^*$ can be assigned an ear level and an s-t number. With the knowledge of the ear level and the s-t number of each node, three independent spanning trees can be constructed accordingly. Without loss of generality, in the sequel, we assume m_1 is a non-cutvertex monitor¹ in $\mathcal{G}$ and virtual node r connects to m_2 , i.e., in the definition of $\mathcal{G}_{ex}^*$ (see Section 4.1), m_i equals m_2 . Due to the special structure of $\mathcal{G}_{ex}^*$, the complicated algorithm [52] of finding $\mathcal{E}_1$ can be avoided, i.e., simply choose $rm_1^+m_1m_2^+r$ (m_1^+ and m_2^+ are virtual monitors) as $\mathcal{E}_1$. Then the rest ears can be found by the ear decomposition algorithm described in the proof of Theorem 1 [52]. To construct 3 independent spanning trees, one rule for selecting the last ear $\mathcal{E}_{n_e}$ is that its internal node must be m_2 .

By running s-t numbering algorithm [64], the final s-t numbers for r and m_1^+ are 1 and k (the number of nodes in $\mathcal{G}_{ex}^*$), respectively. With the computed ear level and s-t numbers of a given 3-vertex-connected graph $\mathcal{G}_{ex}^*$, it is easy to show that each node v in $\mathcal{G}_{ex}^* \setminus \{r, m_1^+, m_2\}$ has three neighbors w_1, w_2 and w_3 such that (i) ear level $g(w_1) > g(v)$, (ii) $f(w_2) < f(v)$ and $g(w_2) \leq g(v)$, and (iii) $f(w_3) > f(v)$ and $g(w_3) \leq g(v)$. Then we construct 3 independent spanning trees as follows:

The above three properties are also the three rules to construct the three independent spanning trees $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ in $\mathcal{G}_{ex}^*$:

- $\mathcal{T}_1$: First, add link rm_2 to $\mathcal{T}_1$. Next, for each node v in $\mathcal{G}_m \setminus \{r, m_2\}$, $\mathcal{T}_1$ involves link vw , where w is a neighbor node of v with $g(w) > g(v)$;
- $\mathcal{T}_2$: First, add link rm_2^+ to $\mathcal{T}_2$. Next, for each node v in $\mathcal{G}_m \setminus \{r, m_2^+\}$, $\mathcal{T}_2$ involves link vw , where w is a neighbor node of v with $f(w) < f(v)$ and $g(w) \leq g(v)$;
- $\mathcal{T}_3$: First, add link rm_1^+ to $\mathcal{T}_3$. Next, for each node v in $\mathcal{G}_m \setminus \{r, m_1^+\}$, $\mathcal{T}_3$

¹A non-cutvertex monitor can be found since it is impossible that all monitors are cutvertices in an identifiable network according to the minimum monitor placement algorithm MMP.

involves link vw , where w is a neighbor node of v with $f(w) > f(v)$ and $g(w) \leq g(v)$.

Graph Decompositions

The basic ideas of biconnected and triconnected component decomposition in [58, 51] are outlined in Algorithms C.1 and C.2. See [58, 51] for the algorithm details.

Algorithm C.1: Biconnected Component Decomposition

input : Connected graph $\mathcal{G}$
output: All biconnected components in $\mathcal{G}$

- 1 each vertex v in $\mathcal{G}$ is assigned a “pre” and a “low” number [58], where the “pre” number is obtained during the depth-first search while the “low” number is computed according to its connections to neighboring vertices;
- 2 identify all cut-vertices based on the “pre” and “low” values;
- 3 store the visited edges in a stack via depth-first search;
- 4 when a cut-vertex is discovered, pop the edges in the stack to get all edges in the same biconnected component;

Algorithm C.2: Triconnected Component Decomposition

input : Connected graph $\mathcal{G}$
output: All triconnected components in $\mathcal{G}$

- 1 partition $\mathcal{G}$ into biconnected components $\mathcal{B}_1, \mathcal{B}_2, \dots$, according to Algorithm C.1;
- 2 **foreach** *biconnected component* $\mathcal{B}_i$ **do**
- 3 find a cycle $\mathcal{C}$ in $\mathcal{B}_i$;
- 4 **if** $\mathcal{C}$ *does not exist* **then**
- 5 $\mathcal{B}_i$ is a triconnected component;
- 6 **else**
- 7 $\mathcal{B}'_i = \mathcal{B}_i - \mathcal{C}$;
- 8 localize all 2-vertex-cuts (see Definition 12) by finding cycles in each connected component within $\mathcal{B}'_i$;
- 9 follow a depth-first search to store the edges belonging to the same triconnected component;
- 10 **end**
- 11 **end**

Supporting Theorems

In this appendix, we first list all supporting theorems in Section D.1 used in this thesis and then give the corresponding proofs in Section D.2.

D.1 Theorems

Lemma D.1. Suppose two monitors are deployed in $\mathcal{G}$ to measure simple paths. If link l is a bridge in $\mathcal{G}$ with one monitor on each side, as illustrated in Fig. D.1, then neither l nor its adjacent links are identifiable.

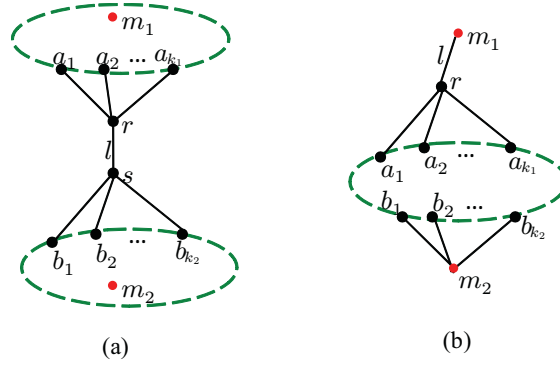


Figure D.1: Two cases of bridge link l : (a) interior bridge, (b) exterior bridge.

Proposition D.2. Using two monitors, the necessary and sufficient condition for $\mathcal{G} + m_1m_2$ being a 3-vertex-connected graph is when 2 nodes are deleted in $\mathcal{G}$, the remaining graph is still connected, *or* every connected component has a monitor.

With the constructed three independent spanning trees ($\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$) in an identifiable network $\mathcal{G}$, let $\mathcal{G}_m := \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$. Then each node in $\mathcal{G}_m \setminus r$ (virtual nodes

m_1^+ and m_2^+ are also included) has three internally vertex disjoint paths (denoted by $\mathcal{X}^{(1)}$, $\mathcal{X}^{(2)}$ and $\mathcal{X}^{(3)}$) to r , each along the corresponding spanning tree. Combining any two of these three paths, we get three cycles, i.e., $\mathcal{C}^{(1)} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)}$, $\mathcal{C}^{(2)} = \mathcal{X}^{(2)} \cup \mathcal{X}^{(3)}$ and $\mathcal{C}^{(3)} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(3)}$. Let C be the set of all these cycles.

Lemma D.3. The number of distinct cycles in C is the number of links (both real and virtual) in $\mathcal{G}_m$, i.e., $|C| = ||\mathcal{G}_m||$. Furthermore, all these distinct cycles are linearly independent.

Claim 1. A triconnected component $\mathcal{T}$ may contain multiple virtual links. For each involved virtual link whose end-points $\{v_1, v_2\}$ (the end-points of a virtual link must form a vertex cut) form a Type-0-VC w.r.t. $\mathcal{T}$, there exists a simple path $\mathcal{P}_r$ with the same end-points in a neighboring biconnected component $\mathcal{B}_{\mathcal{T}}$ which connects to $\mathcal{T}$ via $\{v_1, v_2\}$. $\mathcal{P}_r$ can be used to replace the associated virtual link in $\mathcal{T}$ if this virtual link is chosen to construct measurement paths for identifying real links in $\mathcal{T}$. This replacement operation does not affect all existing path construction policies or the identifiability of real links in $\mathcal{T}$. Such $\mathcal{P}_r$ also exists if $\{v_1, v_2\}$ forms a Type- k -VC ($k \geq 1$), but no agents (excluding v_1 and v_2) in $\mathcal{B}_{\mathcal{T}}$ are used for identifying $\mathcal{T}$.

Lemma D.4. For all vantages $\{\mu_i\}$ in a triconnected component $\mathcal{T}$ of $\mathcal{G}$, there exists an external μ_i -to-agent path $\mathcal{P}$ ($\mathcal{P}$ is a degenerated single node if μ_i is an agent) which is internally vertex disjoint with all other μ_j -to-agent paths ($j \neq i$).

Lemma D.5. Algorithm A.1, Determination of All Identifiable Links in Triangles of Category 2.1, can determine all identifiable links in a triangle of Category 2.1.

Lemma D.6. Algorithm A.2, Determination of All Identifiable Links in Triangles of Category 2.3, can determine all identifiable links in a triangle of Category 2.3.

Lemma D.7. Algorithm 5.3 (in Chapter 5), Determination of Direct Links in Components of Category 3, can determine the identifiability of the direct link in a triconnected component of Category 3.

Theorem D.8. For the triconnected component decomposition of graph $\mathcal{G}$, (i) the 3-vertex-connected and single-link components are unique, while triangle components may not be unique; and (ii) the total number of triconnected components $\mathcal{N}_{\mathcal{T}}$ in $\mathcal{G}$, however, is a constant.

D.2 Proofs

D.2.1 Proof of Lemma D.1

The proof follows similar arguments as in the proof of Theorem 3.1. Consider the case of Fig. D.1 (a). Assume all links except link l and its adjacent links $a_i r$ ($i = 1, \dots, k_1$) and sb_j ($j = 1, \dots, k_2$) are identified. We can then reduce the linear equation associated with any $m_1 \rightarrow m_2$ path to the form

$$W_{a_i r} + W_{sb_j} + W_l = \phi_{ij}, \quad i = 1, \dots, k_1, \quad j = 1, \dots, k_2. \quad (\text{D.1})$$

Writing these equations in matrix form and applying the linear transform in the proof of Theorem III.1 yield a transformed measurement matrix (blank entries are zero):

$$\mathbf{R}' = \left(\begin{array}{c|c|c} W_{a_1 r} \cdots W_{a_{k_1} r} & W_{sb_1} \cdots W_{sb_{k_2}} & W_l \\ \hline \begin{pmatrix} 1 & & \\ 1 & & \\ \vdots & & \\ 1 & & \end{pmatrix} & \begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \\ & & & 1 \end{pmatrix} & \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \\ \hline \begin{pmatrix} & 1 & \\ & & 1 \\ & & & \ddots \\ & & & & 1 \end{pmatrix} & \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} & \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \end{array} \right) \begin{matrix} \left. \begin{matrix} k_2 \\ \text{rows} \end{matrix} \right\} \\ \left. \begin{matrix} k_1 - 1 \\ \text{rows} \end{matrix} \right\} \end{matrix}.$$

This matrix corresponds to the maximum set of linearly independent equations involving the unknown variables W_l , $(W_{a_i r})_{i=1}^{k_1}$, and $(W_{sb_j})_{j=1}^{k_2}$. Since any sub-set of k equations contains more than k unknown variables, none of these variables can be identified. Therefore, l and its adjacent links are all unidentifiable.

In the case of Fig. D.1 (b), similar argument applies, except that $W_{a_i r}$ is replaced by $W_{r a_i}$ and $W_{s b_j}$ is replaced by $W_{b_j m_2}$. $\square$

D.2.2 Proof of Proposition D.2

Necessary part.

1) If $\mathcal{G}$ is separated by deleting 2 non-monitors, then each component must have a monitor; otherwise, $\mathcal{G} + m_1 m_2$ is 2-vertex-connected.

2) If one of the deleted nodes is a monitor, then the remaining graph of $\mathcal{G}$ is the same as that of $\mathcal{G} + m_1 m_2$ when these two nodes are deleted (because link $m_1 m_2$ is deleted). Since $\mathcal{G} + m_1 m_2$ is 3-vertex-connected, this remaining graph must be connected.

3) If m_1 and m_2 are deleted, then the remaining graph is the interior graph $\mathcal{H}$, which is connected according to the assumption.

Sufficient part.

1) If $\mathcal{G}$ is always connected after deleting two nodes, then $\mathcal{G}$ is 3-vertex-connected, so is $\mathcal{G} + m_1 m_2$.

2) If $\mathcal{G}$ is separated into two connected components after deleting two nodes, and each component has a monitor, then adding link $m_1 m_2$ will connect these components again. Therefore, $\mathcal{G} + m_1 m_2$ is 3-vertex-connected. $\square$

D.2.3 Proof of Lemma D.3

For illustrative purpose, we color the links on $\mathcal{T}_1$ as blue, $\mathcal{T}_2$ as green, and $\mathcal{T}_3$ as red (the three independent spanning trees are constructed as stated in Appendix B). Since the independent trees are constructed w.r.t. r , each link in $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ has a natural direction toward r . Therefore, for the simplicity of the following proofs, each link in $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ is assigned a direction toward r . Note this direction is for and only for the theorem proof and only exists on $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, meaning the assumption

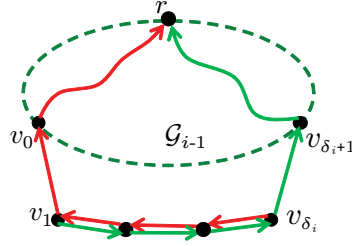


Figure D.2: Paths generated by merging red and green path segments.

that the original graph (see Section 2.2) is undirected still holds. With this notation, links l_1 on $\mathcal{T}_i$ and l_2 on $\mathcal{T}_j$ ($i \neq j$) with different colors and different directions might correspond to the same link in $\mathcal{G}_m$.

In $\mathcal{G}_m$, let b_1 be the number of links appearing only in $\mathcal{T}_1$ (colored as blue), and b_2 be the number of links appearing in two trees: one is $\mathcal{T}_1$ (colored as blue), the other one is either $\mathcal{T}_2$ (colored as green) or $\mathcal{T}_3$ (colored as red). Then we have:

$$b_1 + b_2 = |\mathcal{G}_m| - 1, \quad (\text{D.2})$$

since $\mathcal{T}_1$ is a spanning graph of $\mathcal{G}_m$. Consider ear $\mathcal{E}_i$. Let δ_i be the number of newly added nodes to $\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}$ by $\mathcal{E}_i$. Let $v_0 \dots v_{\delta_i+1}$ denote the nodes in $\mathcal{E}_i$, where v_0 and v_{δ_i+1} are end-points already existed in previous ears. Then we can derive the relationship among the number of links, the number of nodes and the number of ears in $\mathcal{G}_m$. Suppose there are n_e ears in total, then $\mathcal{E}_1$ with δ_1 new nodes has δ_1 links since $\mathcal{E}_1$ is a cycle, whereas $\mathcal{E}_i$ ($2 \leq i \leq n_e$) which is a path with δ_i new nodes has $\delta_i + 1$ new links. In addition to these links, there exist b_1 links only appearing in the blue tree and not involving in any ears. Hence, we have

$$\begin{aligned} ||\mathcal{G}_m|| &= b_1 + \delta_1 + \sum_{i=2}^{n_e} (\delta_i + 1) \\ &= b_1 + \delta_1 + \sum_{i=2}^{n_e} \delta_i + \sum_{i=2}^{n_e} 1 \\ &= b_1 + |\mathcal{G}_m| + n_e - 1. \end{aligned} \quad (\text{D.3})$$

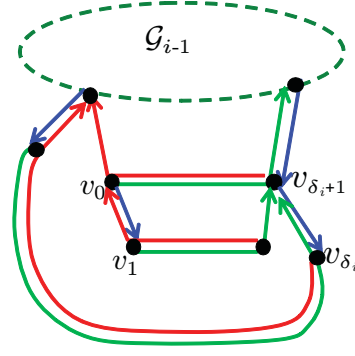


Figure D.3: Paths generated by merging blue and red/green path segments.

To count distinct cycles in C , we consider merging the cycles obtained from each pair of independent spanning trees.

1) We first consider the cycles obtained by combining the red and green paths. For ear $\mathcal{E}_i$, each newly added node in $\mathcal{E}_i$ corresponds to a cycle obtained by combining the red and green paths. However, as Fig. D.2 shows, each link between v_1 and v_{δ_i} corresponds to two colors, red and green, with opposite directions. Therefore, cycles formed by combining the red and green paths w.r.t. $v_1 \cdots v_{\delta_i+1}$ are identical. Thus, ear $\mathcal{E}_i$ only contributes one distinct cycle when combining the red and green paths for any of the newly added nodes $v_1 \cdots v_{\delta_i}$.

2) Next, we consider merging the cycles generated by combining the red and the blue paths. For the δ_i new nodes in ear $\mathcal{E}_i$, each path formed by combining the red and the blue paths are distinct from each other. However, as Fig. D.3 displays, the *red + blue* cycle w.r.t. v_1 might be identical with the *red + blue* cycle w.r.t. v_0 (when v_0 was first added in the previous operations). Thus, the same cycle might be counted twice. Let ε_i indicate if link v_0v_1 is colored as blue, i.e., $\varepsilon_i = 1$ if true, and $\varepsilon_i = 0$ otherwise. With this notation, the number of distinct cycles formed by combining the red and the blue paths is $\delta_i - \varepsilon_i$ for ear $\mathcal{E}_i$ ($2 \leq i \leq n_e$), and $\delta_1 - 1 - \varepsilon_1$ for ear $\mathcal{E}_1$.

3) Finally, we consider merging the cycles formed by combining the green and the blue paths. Following the same argument in 2), each path formed by combining

the green and the blue paths are distinct from the δ_i new nodes in ear $\mathcal{E}_i$. However, the cycles w.r.t. to v_{δ_i} and v_{δ_i+1} might be identical (see Fig. D.3). Let ε'_i indicate if link $v_{\delta_i}v_{\delta_i+1}$ is colored blue, i.e., $\varepsilon'_i = 1$ if true, and $\varepsilon'_i = 0$ otherwise. Accordingly, the number of distinct cycles formed by combining the green and the blue paths is $\delta_i - \varepsilon'_i$ for ear $\mathcal{E}_i$ ($2 \leq i \leq n_e$), and $\delta_1 - 1 - \varepsilon'_1$ for ear $\mathcal{E}_1$.

Therefore, the number of distinct cycles, denoted by Q_i , contributed by ear $\mathcal{E}_i$ is:

$$\begin{aligned} Q_1 &= 1 + \delta_1 - 1 - \varepsilon_1 + \delta_1 - 1 - \varepsilon'_1, \\ Q_i &= 1 + \delta_i - \varepsilon_i + \delta_i - \varepsilon'_i \quad (2 \leq i \leq n_e). \end{aligned} \tag{D.4}$$

Thus, the number of distinct cycles in C is

$$\begin{aligned} |C| &= Q_1 + \sum_{i=2}^{n_e} Q_i \\ &= 2\delta_1 - \varepsilon_1 - \varepsilon'_1 - 1 + 2 \sum_{i=2}^{n_e} \delta_i + \sum_{i=2}^{n_e} 1 - \sum_{i=2}^{n_e} (\varepsilon_i + \varepsilon'_i) \\ &= 2|\mathcal{G}_m| + n_e - 2 - \sum_{i=1}^{n_e} (\varepsilon_i + \varepsilon'_i) \\ &= 2|\mathcal{G}_m| + n_e - 2 - b_2 \end{aligned} \tag{D.5}$$

Subtracting (D.3) from (D.5) and utilizing (D.2), we get

$$|C| = \|\mathcal{G}_m\|. \tag{D.6}$$

Let $\mathcal{G}'_m := \mathcal{G}_m$ except all virtual links/nodes in $\mathcal{G}_m$ are real links/nodes in $\mathcal{G}'_m$. Suppose cycle measurement is allowed and r is the only monitor in $\mathcal{G}'_m$, then all cycle measurements associated with C are sufficient to identify all links in $\mathcal{G}'_m$ (following the same method in STLI, see Section 4.3). Moreover, we have $\|\mathcal{G}_m\| = \|\mathcal{G}'_m\|$; therefore, all cycles in C are linearly independent. $\square$

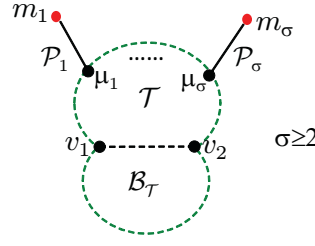


Figure D.4: Virtual link replacement.

D.2.4 Proof of Claim 1

Fig. D.4 illustrates a triconnected component $\mathcal{T}$ with σ vantages ($\sigma \geq 2$), where some of these σ vantages may form conjugate pairs (see Definition 14 in Appendix A). For vertex cut $\{v_1, v_2\}$ (which is a Type-0-VC), there exists a simple path $\mathcal{P}_r$ connecting v_1 and v_2 in the neighboring biconnected component $\mathcal{B}_{\mathcal{T}}$ within the parent biconnected component of $\mathcal{T}$ as $\mathcal{B}_{\mathcal{T}}$ contains at least 3 nodes. We know that $\mathcal{B}_{\mathcal{T}}$ connects to $\mathcal{T}$ via only $\{v_1, v_2\}$; therefore, $\mathcal{P}_1, \dots, \mathcal{P}_{\sigma}$ do not have common nodes with $\mathcal{B}_{\mathcal{T}}$ except that μ_i ($i = 1, \dots, \sigma$) may equal v_1 (v_2). Hence, for virtual link $v_1 v_2$, if it is used for identifying real links in $\mathcal{T}$ according to Theorem 3.3 or 3.7, then it can be replaced by $\mathcal{P}_r$ which is a simple path and can be abstracted as a real link in $\mathcal{T}$.

For the case that $\{v_1, v_2\}$ forms a Type- k -VC ($k \geq 1$) and no agents (excluding v_1 and v_2) in $\mathcal{B}_{\mathcal{T}}$ are used for identifying $\mathcal{T}$, we can ignore these agents in $\mathcal{B}_{\mathcal{T}}$. Accordingly, this Type- k -VC is converted to a Type-0-VC. Therefore, the above argument applies to this special case, suggesting the existence of replacement path $\mathcal{P}_r$. $\square$

D.2.5 Proof of Lemma D.4

We consider the following four cases.

1) If vantage μ_i is an agent, then μ_i -to-agent path is itself μ_i (i.e., a path containing only one node). This is a trivial case that μ_i -to-agent path is internally vertex

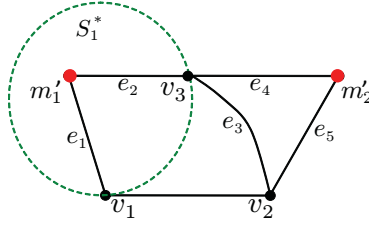
disjoint with all other vantage-to-agent paths.

2) For triconnected component $\mathcal{T}$ in biconnected component $\mathcal{B}$, if vantage μ_i is not an agent and μ_i is in a Type-1-VC, then let μ'_i be the other node in this Type-1-VC, $\mathcal{P}$ the μ_i -to-agent path, and $\mathcal{P}'$ the μ'_i -to-agent path. If μ_i and μ'_i form a conjugate pair (see Definition 14 in Appendix A), then $\mathcal{P}$ and $\mathcal{P}'$ may have a common terminating-point (i.e., the agent). In this case, in addition to the common terminating-point m' , if $\mathcal{P}$ and $\mathcal{P}'$ must have another common node w (an internal vertex in $\mathcal{P}$ and $\mathcal{P}'$), then it implies that μ_i and μ'_i cannot connect to m' if w is deleted. Therefore, w is a cut-vertex in $\mathcal{B}$, contradicting the assumption that $\mathcal{B}$ is a biconnected component. If μ_i and μ'_i do not form a conjugate pair, then μ'_i is an agent. In this case, μ'_i -to-agent path is μ'_i , which is internally vertex disjoint with $\mathcal{P}$.

3) For triconnected component $\mathcal{T}$ in biconnected component $\mathcal{B}$, if vantage μ_i is not an agent and μ_i is in a Type- k -VC ($k \geq 2$), then we can always construct μ_i -to- m' path $\mathcal{P}$ and μ'_i -to- m' path $\mathcal{P}'$ (μ'_i is another node in this Type- k -VC, suppose μ'_i is not an agent) to make sure that $\mathcal{P}$ and $\mathcal{P}'$ have the same terminating-point m' . Then the same argument in 2) applies to this case.

4) In 2) and 3), we only consider constructing $\mathcal{P}$ and $\mathcal{P}'$ in the same neighboring biconnected component $\mathcal{B}_N$ of $\mathcal{T}$. For the vantage-to-agent paths in other neighboring biconnected component of $\mathcal{T}$, these paths do not use any nodes in $\mathcal{B}_N$ (except for the possible case that the starting-point, i.e., the vantage, is in $\mathcal{B}_N$). Thus, these vantage-to-agent paths are still internally vertex disjoint with $\mathcal{P}$.

Therefore, for vantages in the same Type- k -VC, their vantage-to-agent paths at most have one common terminating-point (i.e., the agent). While for vantages in different Type- k -VCs, their vantage-to-agent paths at most have a common starting-point (i.e., the vantage). Consequently, For all vantages $\{\mu_i\}$ in $\mathcal{T}$, there exists an external μ_i -to-agent path $\mathcal{P}$ which is internally vertex disjoint with all other μ_j -to-agent paths ($j \neq i$). $\square$

Figure D.5: Triangle identification (v_1 is an agent).

D.2.6 Proof of Lemma D.5

For real link l_i in triconnected component $\mathcal{T}$, let v_1 and v_2 be the end-points of l_i , v_3 the third node in $\mathcal{T}$, S_1 (S_2) the set of immediately neighboring triconnected components connected to $\mathcal{T}$ via $\{v_1, v_3\}$ ($\{v_2, v_3\}$), and S_1^* (S_2^*) the set of immediately neighboring biconnected components connected to $\mathcal{T}$ via $\{v_1, v_3\}$ ($\{v_2, v_3\}$) within the same parent biconnected component. Note all links in S_1^* (S_2^*) except for v_1v_3 (v_2v_3) are real links. It suffices to show how the identifiability of l_i is determined by Algorithm A.1. Let $\mathcal{T}_1^*$ ($\mathcal{T}_2^*$) be a neighboring triconnected component of $\mathcal{T}$ with the following properties: (i) $v_1 \in \mathcal{T}_1^*$ ($v_2 \in \mathcal{T}_2^*$); (ii) $v_2 \notin \mathcal{T}_1^*$ ($v_1 \notin \mathcal{T}_2^*$); (iii) $\mathcal{T}_1^*$ ($\mathcal{T}_2^*$) contains at least two real links which are incident to v_1 (v_2); (iv) $\mathcal{T}_1^*$ ($\mathcal{T}_2^*$) contains 3 or more vantages.

We first consider the condition (called Condition A in the sequel) that: (a) v_1v_3 is virtual **AND** (b) v_1 is not an agent in the parent biconnected component of $\mathcal{T}$ **AND** (c) $\{v_1, v_2\}$ is a Type-0-VC w.r.t. $\mathcal{T}$ if $\{v_1, v_2\}$ is a 2-vertex cut **AND** (d) $|S_1| = 1$ **AND** (e) $\mathcal{T}_1^*$ satisfying the properties (i)–(iv) does not exist. We show that if $\mathcal{T}$ does not satisfy any of the five conditions (a)–(e) in Condition A, then there exists a path replacement for v_1v_3 (denoted by $\mathcal{P}(v_1, v_r)$), one v_1 -to-agent path, and one v_r -to-agent path, with all these three paths being internally vertex disjoint.

1) If v_1v_3 is a real link, then it is obvious that there is no need to find a path replacement for v_1v_3 . Moreover, according to the definition of Category 2.1 (see Appendix A), there exist two vantage-to-agent (i.e., v_1 -to-agent and v_3 -to-agent) disjoint paths.

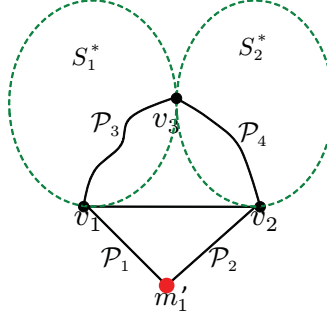


Figure D.6: Triangle identification (v_1v_2 is a cross-link).

2) If v_1 is an agent in the parent biconnected component of $\mathcal{T}$, then v_1 itself is a degenerated v_1 -to-agent path (containing a single node). In this case, if S_1^* contains one agent (excluding v_1), say m_1' (as illustrated in Fig. D.5), then there exist internally vertex disjoint paths $m_1'e_1v_1$ and $m_1'e_2v_3$ since each component in S_1^* is 2-vertex-connected. Hence, $m_1'e_1v_1$ can be used as the path replacement for v_1v_3 . Note that $m_1'e_2v_3$ might be merged with another path for identifying v_1v_2 , which will be clear in later discussions. If S_1^* does not contain any agents except for v_1 , then there exists path replacement connecting v_1 and v_3 within S_1^* . Furthermore, for this replacement path, the end-point v_3 has a vantage-to-agent path in S_2^* or v_3 itself is an agent according to the definition of Category 2.1.

3) If $\{v_1, v_2\}$ is a Type- k -VC ($k \geq 1$) w.r.t. $\mathcal{T}$ when $\{v_1, v_2\}$ is a 2-vertex cut, then, as illustrated in Fig. D.6, v_1v_2 is a Cross-link (defined in Section 3.2.2.1), which is identifiable using paths $\mathcal{P}_1, \dots, \mathcal{P}_4$.

4) If $|S_1| > 1$, then it implies $|S_1^*| > 1$. Thus, the path replacement for v_1v_3 can be chosen from one component in S_1^* and there exist v_1 -to-agent (v_3 -to-agent) path in another component of S_1^* .

5) Suppose there exists $\mathcal{T}_1^*$ satisfying the properties in (i)–(iv). This is the most complicated condition in Condition A. Let $\mathcal{B}$ denote the parent biconnected component of $\mathcal{T}$. If we remove the neighboring biconnected component (except for nodes v_1 and v_2) connecting to $\mathcal{T}$ via v_1v_2 , then, as illustrated in Fig. D.7, v_3 is a cut-vertex in the remaining graph $\mathcal{B}^*$. In $\mathcal{B}^*$, for the connected component containing v_1 and

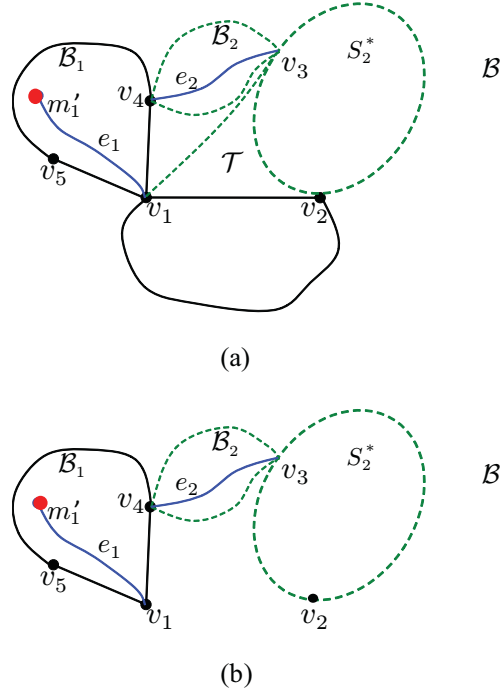


Figure D.7: Triangle identification (there exists $\mathcal{T}_1^*$ in $\mathcal{B}_1$).

v_3 , there exists a sub-graph $\mathcal{B}_1$ which is 2-vertex-connected or a bond¹, as illustrated in Fig. D.7. Note that in Fig. D.7, all links in $\mathcal{B}_1$ and $\mathcal{B}_2$ are real links since they are 2-vertex-connected (or bonds), and $\mathcal{B}_1$ contains $\mathcal{B}_2$ when v_3 is the only cut-vertex in $\mathcal{B}^*$. Suppose (a)–(d) in Condition A are satisfied. If there exist real links v_1v_4 and v_1v_5 in $\mathcal{B}_1$, then $\mathcal{T}_1^*$ must be a sub-graph in $\mathcal{B}_1$. Since v_2 and v_3 are vantages of $\mathcal{T}$, v_1 and v_4 must be the two vantages of $\mathcal{T}_1^*$. If $\mathcal{T}_1^*$ contains a third vantage, then it implies that there exists an agent m'_1 (with $m'_1 \neq v_1, m'_1 \neq v_4$) in $\mathcal{B}_1$. Hence, as $\mathcal{B}_1$ is 2-vertex-connected, there exist internally vertex disjoint paths $v_1e_1m'_1$ and $v_1v_4e_2v_3$. Therefore, $v_1v_4e_2v_3$ is chosen as the path replacement for v_1v_3 . Moreover, there exist v_3 -to-agent path (guaranteed by Condition B discussed as follows) in S_2^* or v_3 itself is an agent in $\mathcal{B}$.

Let Condition B be (a) v_2v_3 is virtual **AND** (b) v_2 is not an agent in the parent biconnected component of $\mathcal{T}$ **AND** (c) $\{v_1, v_2\}$ is a Type-0-VC w.r.t. $\mathcal{T}$ if $\{v_1, v_2\}$ is a 2-vertex cut **AND** (d) $|S_2| = 1$ **AND** (e) $\mathcal{T}_2^*$ satisfying the properties in (i)–(iv)

¹A bond is a graph with only two nodes and one link connecting these two nodes.

does not exist. The above argument in 1)–5) also applies to the path replacement for v_2v_3 . If neither Condition A nor Condition B is satisfied, then the two path replacements for v_1v_3 and v_2v_3 can be abstracted as two single links, thus forming a triangle containing link v_1v_2 . Moreover, there exist vertex disjoint vertex-to-agent paths for the three vertices in this constructed triangle. Therefore, v_1v_2 is identifiable since the resided triangle with three vertex-to-agent paths (abstracting paths as single links) satisfies the condition in Theorem 3.7.

Remark: When neither Condition A nor Condition B is satisfied, there are two other possible scenarios. (i) As shown in Fig. D.5, if the replacement path for v_2v_3 is $v_2e_3v_3$, then $m'_1e_2v_3 + v_3e_3v_2$ can be abstracted as a single link in a triangle; (ii) In the case that v_1 and v_2 are both agents, if the replacement path for v_2v_3 is $m'_2e_5v_2$ or v_3 itself is an agent, then a quadrangle (i.e., graph $(\{m'_1, v_1, v_2, m'_2\}, \{m'_1e_1v_1, v_1v_2, v_2e_5m'_2, m'_1v_3m'_2\})$ or graph $(\{m'_1, v_1, v_2, v_3\}, \{m'_1e_1v_1, v_1v_2, v_2e_3v_3, m'_1e_2v_3\})$, where paths are abstracted as single links) is formed. Nevertheless, this quadrangle with its vertex-to-agent paths (abstracting paths as single links) also satisfies the condition in Theorem 3.7.

If (a)–(e) in Condition A are satisfied, then it implies that $|S_1^*| = 1$ and there is no agent (except for v_4) in $\mathcal{B}_1$ (see Fig. D.7). Therefore, we can at most identify the sum metric of v_1v_2 and a path within $\mathcal{B}_1$, but not the individual link metric on v_1v_2 . Same argument applies when (a)–(e) in Condition B are satisfied. Therefore, the identifiability of l_i (with end-points v_1 and v_2) of Category 2.1 can be determined by Algorithm A.1. $\square$

D.2.7 Proof of Lemma D.6

Category 2.3 is defined in Appendix A. Let the two conjugate pairs (see Definition 14) be $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_3\}$ and S_1 (S_2) the set of immediately neighboring triconnected components connecting to $\mathcal{T}$ via $\{\mu_1, \mu_2\}$ ($\{\mu_1, \mu_3\}$), as illustrated in

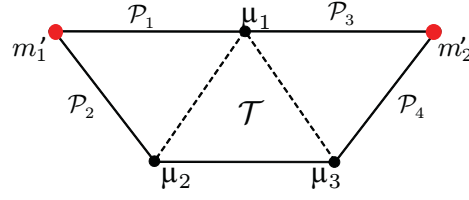


Figure D.8: Link identifications of Category 2.3.

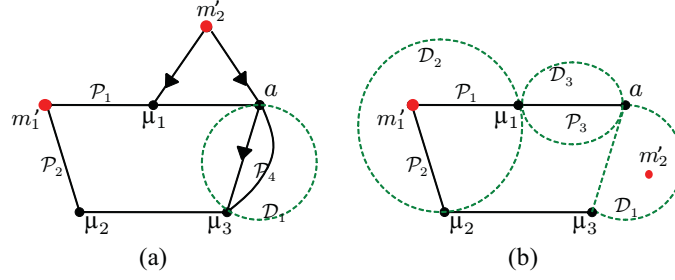
Figure D.9: Category 2.3 - unidentifiable $\mu_2\mu_3$.

Fig. D.8. Since $\{\mu_1, \mu_2\}$ and $\{\mu_1, \mu_3\}$ are 2-vertex-cuts, we know that there exist internally vertex disjoint paths $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ and $\mathcal{P}_4$. If $\mu_1\mu_2$ (or $\mu_1\mu_3$) is a real link, then $\mu_1\mu_2$ (or $\mu_1\mu_3$) is known as a Cross-link (defined in Section 3.2.2.1) since $\mu_2\mu_3$ can be replaced by a path in neighboring biconnected components if $\mu_2\mu_3$ is virtual. Therefore, $\mu_1\mu_2$ (or $\mu_1\mu_3$) is identifiable. Now we focus on the identification of $\mu_2\mu_3$ in $\mathcal{T}$ (when $\mu_2\mu_3$ is a real link). The conditions to guarantee the identifiability of $\mu_2\mu_3$ is:

(link $\mu_1\mu_2$ is real **OR** $|S_1| \geq 2$ **OR** one component in S_1 is 3-vertex-connected) **AND**
 (link $\mu_1\mu_3$ is real **OR** $|S_2| \geq 2$ **OR** one component in S_2 is 3-vertex-connected).

Suppose the above condition is not satisfied. Then we can prove $\mu_2\mu_3$ is unidentifiable as follows.

We first consider the condition: link $\mu_1\mu_3$ is real **OR** $|S_2| \geq 2$ **OR** one component in S_2 is 3-vertex-connected. If not satisfied, then it implies that there is no real link $\mu_1\mu_3$ and the only immediately neighboring triconnected component is a triangle, i.e., $\mu_1-a-\mu_3$ in Fig. D.9 (μ_1a and μ_3a can be virtual links as well).

1) If agent m'_2 is in the location shown in Fig. D.9 (a), then all paths from m'_1 to m'_2 traversing $\mu_2\mu_3$ must use one simple path in $\mathcal{D}_1$. Therefore, the best case is that

we can compute the sum metric of link $\mu_2\mu_3$ and another link which is incident to μ_3 in $\mathcal{D}_1$, but cannot compute them separately.

2) If agent m'_2 is in the location shown in Fig. D.9 (b), then $\mathcal{P}_3$ and $\mu_2\mu_3$ become a “double bridge” connecting $\mathcal{D}_2$ and $\mathcal{D}_1$. Abstracting $\mathcal{P}_3$ as a single link, Section 3.2.1 proves that none of the links in a double bridge is identifiable when constraining the measurement paths to simple paths. If we choose other paths as $\mathcal{P}_3$ in $\mathcal{D}_3$, then the same argument applies. Therefore, based on 1) and 2), $\mu_2\mu_3$ is unidentifiable.

Analogously, we can prove that $\mu_2\mu_3$ is unidentifiable when condition (link $\mu_1\mu_2$ is real **OR** $|S_1| \geq 2$ **OR** one component in S_1 is 3-vertex-connected) is not satisfied.

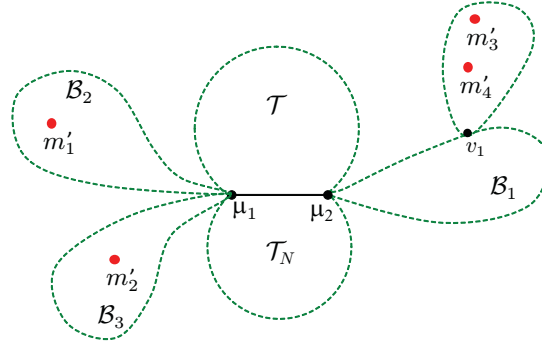
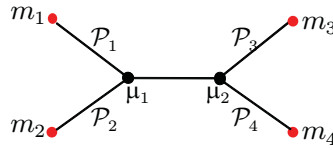
When the required conditions are satisfied, we can prove that $\mu_2\mu_3$ is identifiable as follows:

If $\mu_1\mu_2$ ($\mu_1\mu_3$) is a virtual link, then it can be replaced by a path in a neighboring component. For instance, if $|S_1| \geq 2$, then one replacement path can be found in one component of S_1 . If one component in S_1 is 3-vertex-connected, then there exist 2 internally vertex disjoint paths (each with the number of links greater than 1) connecting μ_1 and μ_2 . Thus, we can choose one of them as a replacement path. Note that the virtual links possibly involved in the replacement paths can be further replaced by the paths in their neighboring components within the same parent biconnected component. After these replacement operations, $\mu_2\mu_3$ in Fig. D.8 is a Shortcut (defined in Section 3.2.2.1), which is proved to be identifiable.

Therefore, Algorithm A.2 can determine all identifiable/unidentifiable links in a triangle triconnected component of Category 2.3. $\square$

D.2.8 Proof of Lemma D.7

For direct link $\mu_1\mu_2$ in $\mathcal{T}$, if there exist neighboring triconnected components connecting to $\mathcal{T}$ by 2-vertex cut $\{\mu_1, \mu_2\}$ **AND** μ_1 and μ_2 are not the (only two) vantages in one of these neighboring components $\mathcal{T}_N$, then $\mu_1\mu_2$ is not a direct link connecting

Figure D.10: Direct link $\mu_1\mu_2$ identification.Figure D.11: Direct link $\mu_1\mu_2$ identification in the abstracted graph.

two vantages in $\mathcal{T}_N$. Therefore, if $\mathcal{T}_N$ is of Category 1 or 2, then Algorithm DAIL can determine the identifiability of $\mu_1\mu_2$; if $\mathcal{T}_N$ is of Category 3, then $\mu_1\mu_2$ is identifiable since it is not a direct link in $\mathcal{T}_N$. The above argument is complete, since the number of vantages in $\mathcal{T}_N$ cannot be less than 2 as the parent biconnected component of $\mathcal{T}$ (and $\mathcal{T}_N$) contains at least 2 agents.

If the condition in line 1 of Algorithm 5.3 is not satisfied, then there is no neighboring triconnected component connecting to $\mathcal{T}$ by 2-vertex cut $\{\mu_1, \mu_2\}$, or all these neighboring components contain only μ_1 and μ_2 as vantages, shown in Fig. D.10. In this case, the identifiability of $\mu_1\mu_2$ can be determined with the assistance of neighboring biconnected components. Let $\mathcal{B}_1$ be the only biconnected component connecting to $\mathcal{T}$ via μ_2 , as shown in Fig. D.10. Suppose μ_2 is not a real monitor and v_1 is the only agent (except for μ_2) in $\mathcal{B}_1$. Then we can at most identify the sum metric of $\mu_1\mu_2$ and a path connecting μ_2 and v_1 in $\mathcal{B}_1$, but not the individual link metric on $\mu_1\mu_2$. Therefore, if $\mathcal{B}_1$ is the only biconnected component connecting to $\mathcal{T}$ by μ_2 and μ_2 is not a real monitor, then $\mathcal{B}_1$ must contain at least 2 agents (except for μ_2) for identifying $\mu_1\mu_2$. The same argument applies to the biconnected components connecting to $\mathcal{T}$ via μ_1 . Suppose none of μ_1 or μ_2 is a real monitor. If there are more

than one biconnected component connecting to $\mathcal{T}$ via μ_1 (or μ_2), e.g., $\mathcal{B}_2$ and $\mathcal{B}_3$ in Fig. D.10, then the total number of agents (except for μ_1 (or μ_2)) must be no less than 2 for identifying $\mu_1\mu_2$. If these required conditions regarding neighboring biconnected components of μ_1 and μ_2 are satisfied, then this scenario can be abstracted as Fig. D.11, which satisfies the condition (suppose μ_1 and μ_2 are not monitors) in Theorem 3.7, and thus $\mu_1\mu_2$ is identifiable. Note that for node μ_1 , if μ_1 is a real monitor, then there is no requirement for the neighboring biconnected components of μ_1 , since the sub-graph containing $\mu_1\mu_2$, $\mathcal{P}_3$, $\mathcal{P}_4$ and their associated end-points already satisfies the condition in Theorem 3.7. The same argument applies to the case when μ_2 is a real monitor. Consequently, Algorithm 5.3 can determine the identifiability of $\mu_1\mu_2$ when the condition in line 1 is not satisfied, whereas the case satisfying the condition in line 1 can be determined in neighboring triconnected components. $\square$

D.2.9 Proof of Theorem D.8

Definition 17. *merge*: In graph $\mathcal{G}$, two triconnected components $\mathcal{T}_1$ and $\mathcal{T}_2$ with a common virtual link ab , i.e., $ab \in L(\mathcal{T}_1) \cap L(\mathcal{T}_2)$, can be merged. The resulting graph after the merging operation is $\mathcal{G} = (V', L')$, where $V' = V(\mathcal{T}_1) \cup V(\mathcal{T}_2)$, $L' = L(\mathcal{T}_1) \cup L(\mathcal{T}_2) \setminus \{ab\}$.

1) In [51], it has proved (see Lemma 2 in [51]) that 3-vertex-connected and single-link components in a graph decomposition are unique. Moreover, Lemma 2 in [51] also proved that if we merge (see Definition 17) all triangle components as much as possible, then a set of polygons² are obtained and these polygons are unique. Since there are multiple ways to decompose a polygon, the generated triangle components may not be unique.

2) To prove the total number of triconnected components $\mathcal{N}_{\mathcal{T}}$ in $\mathcal{G}$ is fixed, it suffices to show that the total number of triangle components is fixed regardless of

²A polygon is a connected graph with each node connecting two and only two neighbors.

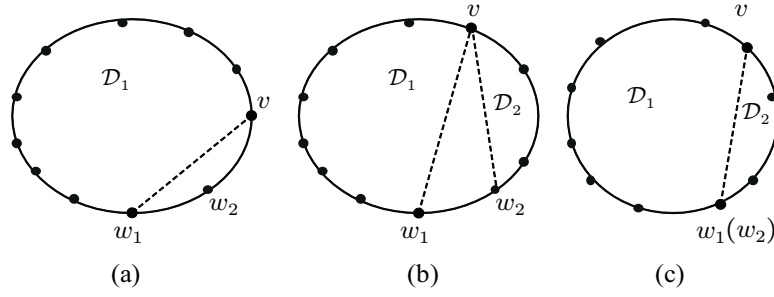


Figure D.12: Decomposition of triconnected components in a polygon.

the mechanisms being used for graph decomposition, since all other types of triconnected components are unique as proved in *I*). To this end, we have one claim: for a unique polygon $\mathcal{D}$ obtained in *I*) with τ nodes ($\tau \geq 3$), the number of triangles after graph decomposition on $\mathcal{D}$ is always $\tau - 2$, which is irrespective of decomposition strategies. We prove this claim by induction. (1) If $\tau = 3$, then there is no need to further decompose $\mathcal{D}$, since $\mathcal{D}$ is already a triangle. Hence, in this case, the number of triangles is only one, i.e., $\tau - 2$. (2) Suppose the claim is true for $\tau = k$ (k is an integer, $k \geq 3$). (3) Consider the case that $\tau = k + 1$. Fig. D.12 (a)–(b) display a polygon with $k + 1$ nodes. For real link w_1w_2 on this polygon, after decomposition, it must belong to one triangle $\mathcal{T}$. Let the third node on $\mathcal{T}$ be v (see Fig. D.12 (a)–(b)). Then there are two cases for $\mathcal{T}$: (i) vw_2 (or vw_1) is a real link in the original graph (Fig. D.12 (a)). In this case, excluding triangle $\mathcal{T}$, the remaining graph is polygon $\mathcal{D}_1$ with virtual link vw_1 (or vw_2). Then to get all triangles, $\mathcal{D}_1$ needs to be further decomposed. We know $|V(\mathcal{D}_1)| = k$, and thus it corresponds to $k - 2$ triangles according to the hypothesis in (2). Therefore, the total number of triangles is $(k + 1) - 2$; (ii) neither vw_1 nor vw_2 is a real link in the original graph $\mathcal{D}$. Then excluding triangle $\mathcal{T}$, the remaining graphs are $\mathcal{D}_1$ and $\mathcal{D}_2$ (Fig. D.12 (b)). To further decompose $\mathcal{D}_1$ and $\mathcal{D}_2$, we can first combine $\mathcal{D}_1$ and $\mathcal{D}_2$ as follows: Let $w_1 = w_2$. The graph combination $\mathcal{D}'$ is the union of $\mathcal{D}_1$ and $\mathcal{D}_2$, excluding virtual link vw_1 (vw_2) (see Fig. D.12 (c)). Then the decomposition of $\mathcal{D}_1$ and $\mathcal{D}_2$ is equivalent to the decomposition of $\mathcal{D}'$, conditioned on that the vertex-cut $\{v, w_1\}$ (the same

as $\{v, w_2\}$) must be used. We know the generated $\mathcal{D}'$ is a polygon, which contains k nodes, and thus it always corresponds to $k - 2$ triangles (including the case that vertex-cut $\{v, w_1\}$ ($\{v, w_2\}$) must be used). Therefore, original graph $\mathcal{D}$ corresponds to $(k + 1) - 2$ triangles, completing the proof. $\square$

Theorem Proof

E.1 Proof of Lemma 3.4

Let m_1^* and m_2^* be two nodes with $m_1^*, m_2^* \in \{m_1, m_2\}$ and $m_1^* \neq m_2^*$ (i.e., $m_1^* = m_1, m_2^* = m_2$ or $m_1^* = m_2, m_2^* = m_1$). In the sequel, Conditions ①–② refer to the two conditions in Theorem 3.3.

1) In $\mathcal{G}$, we first prove the existence of two cycles $\mathcal{C}_1$ and $\mathcal{C}_2$ with $vw \in L(\mathcal{C}_1)$, $vw \in L(\mathcal{C}_2)$ and $|V(\mathcal{C}_1) \cap V(\mathcal{C}_2)| = 2$ or 3.

(1.i). *Existence of $\mathcal{C}_1$.* For $vw \in L(\mathcal{H})$, $\exists$ an H-path¹ $\mathcal{P}_{vw}$ from v to w in a 2-vertex-connected graph, according to Proposition 3.1.3 [47] ($\mathcal{G}$ is a 2-vertex-connected graph, since $\mathcal{G} + m_1m_2$ is 3-vertex-connected). Thus, a cycle $\mathcal{C}_1 = \mathcal{P}_{vw} + vw$ is formed.

(1.ii). *Existence of $\mathcal{C}_2$.* Suppose $\mathcal{C}_2$ does not exist in $\mathcal{G}$. Then to connect v and w , besides using link vw , $\mathcal{C}_1 - vw$ is the only alternative way. Hence, each link in $\mathcal{C}_1 - vw$ must be a bridge in $\mathcal{G} - vw$, contradicting Condition ①. Therefore, there exist $\mathcal{C}_1$ with $vw \in L(\mathcal{C}_1)$ and $\mathcal{C}_2$ with $vw \in L(\mathcal{C}_2)$ ($\mathcal{C}_1 \neq \mathcal{C}_2$).

(1.iii). *By contradiction, we prove it is impossible that $\mathcal{C}_1$ and $\mathcal{C}_2$ must have more than one common link, i.e., $L(\mathcal{C}_1) \cap L(\mathcal{C}_2) = \{vw\}$.* Suppose $\mathcal{C}_2$ must share some unavoidable² common links with $\mathcal{C}_1 \setminus \{vw\}$. Let rs be one of these common links (this is the special case that $\mathcal{C}_1$ and $\mathcal{C}_2$ have more than three common nodes). Then

¹ $\mathcal{P}$ ($||\mathcal{P}|| \geq 1$) is an H-path of graph $\mathcal{H}$ if $\mathcal{P}$ meets $\mathcal{H}$ exactly in its end-points.

²In this proof, *unavoidable* nodes/links are the common nodes/links between $\mathcal{C}_1$ and $\mathcal{C}_2$ in $\mathcal{G}$. These common nodes/links exist regardless of how $\mathcal{C}_1$ and $\mathcal{C}_2$ are selected.

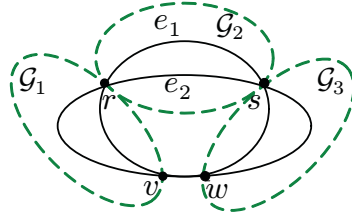


Figure E.1: Two cycles with four common nodes.

if vw is deleted, all possible paths connecting v and w must traverse link rs . In this case, rs becomes a bridge, contradicting Condition ①.

(1.iv). *By contradiction, we prove it is impossible that $\mathcal{C}_1$ and $\mathcal{C}_2$ must have more than three common nodes no matter how $\mathcal{C}_1$ and $\mathcal{C}_2$ are selected, i.e., $\mathcal{C}_1$ and $\mathcal{C}_2$ with $|V(\mathcal{C}_1) \cap V(\mathcal{C}_2)| \leq 3$ can be found.*

We have $\min(|V(\mathcal{C}_1) \cap V(\mathcal{C}_2)|) = 2$, i.e., $\mathcal{C}_1$ and $\mathcal{C}_2$ have at least two common nodes v and w , since $vw \in L(\mathcal{C}_1)$ and $vw \in L(\mathcal{C}_2)$. Suppose there are always four common nodes between $\mathcal{C}_1$ and $\mathcal{C}_2$ no matter what strategy is used to select the two cycles. Let $r, s \in V(\mathcal{C}_1) \cap V(\mathcal{C}_2) \setminus \{v, w\}$ (shown in Fig. E.1) denote the two unavoidable common nodes. Without utilizing link vw , to connect v and w , all paths leaving from v and terminating at w must first traverse r and then traverse s . Therefore, $\mathcal{G}$ can be reformed as sub-graphs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ (shown in Fig. E.1) with $V(\mathcal{G}_1 \cap \mathcal{G}_2) = r$, $V(\mathcal{G}_2 \cap \mathcal{G}_3) = s$, and $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 + vw$. It has been proved in (1.iii) that $\mathcal{C}_1$ and $\mathcal{C}_2$ cannot have unavoidable common links apart from vw , thus each of $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ has at least three nodes. Since at most two sub-graphs, say $\mathcal{G}_1$ and $\mathcal{G}_2$, can contain monitors, nodes in $\mathcal{G}_3 - s - w$ are disconnected to monitors, contradicting Proposition D.2. Then obviously, $\mathcal{C}_1$ and $\mathcal{C}_2$ cannot have more than four unavoidable common nodes. Therefore, with the flexibility of selecting all possible $\mathcal{C}_1$ and $\mathcal{C}_2$, it is impossible that $\mathcal{C}_1$ and $\mathcal{C}_2$ must have more than three common nodes.

2) With all possible selections, now we prove $\mathcal{C}_1$ and $\mathcal{C}_2$ can be selected to ensure that one of them, say $\mathcal{C}_1$, is a non-separating cycle and the properties of $vw \in L(\mathcal{C}_1)$, $vw \in L(\mathcal{C}_2)$ and $|V(\mathcal{C}_1) \cap V(\mathcal{C}_2)| = 2$ or 3 can still be preserved.

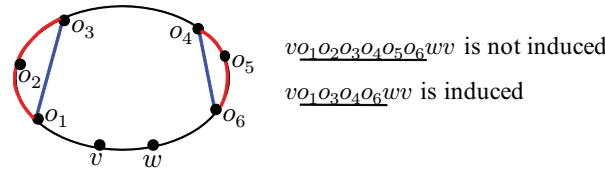
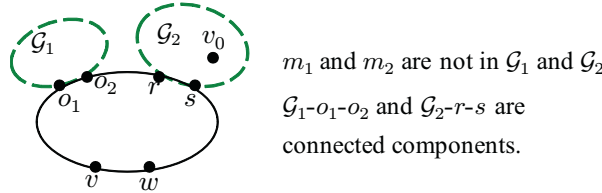


Figure E.2: Replacement operation to obtain an induced cycle.

Figure E.3: Counter example - a non-separating cycle containing vw cannot be found.

(2.i). We prove some links on $\mathcal{C}_1$ with $|V(\mathcal{C}_1) \cap V(\mathcal{C}_2)| = 2$ or 3 can be replaced to ensure the resulting $\mathcal{C}_1$ is an induced graph. If $xy \in L(\mathcal{G})$ with $x, y \in V(\mathcal{C}_1)$ and $xy \notin L(\mathcal{C}_1)$, then use xy to replace $\mathcal{P}_{\mathcal{C}_1}(x, y)$ recursively, i.e., $\mathcal{C}_1 = \mathcal{C}_1 \setminus \overset{\circ}{\mathcal{P}_{\mathcal{C}_1}}(x, y) + xy$ (see Mathematical Notations), until no such xy exists, where $\mathcal{P}_{\mathcal{C}_1}(x, y)$ is the path from x to y in $\mathcal{C}_1$ with $vw \notin L(\mathcal{P}_{\mathcal{C}_1}(x, y))$ (vw is the common link between $\mathcal{C}_1$ and $\mathcal{C}_2$). Finally, $\mathcal{C}_1$ is an induced cycle. Note for each replacement operation, link xy does not belong to $\mathcal{C}_2$ since $\mathcal{C}_1$ and $\mathcal{C}_2$ only have 2 or 3 common nodes (two of these two common nodes are v and w). Therefore, this replacement operation on $\mathcal{C}_1$ does not affect the number of common nodes between $\mathcal{C}_1$ and $\mathcal{C}_2$. This process is shown in Fig. E.2, where all red segments are replaced by the blue segments.

(2.ii). We prove that the induced graph $\mathcal{C}_1$ can be selected to ensure $\mathcal{C}_1$ is a non-separating cycle. Suppose not. Then for all possible cycles containing vw and satisfying (a) in Lemma 3.4, none of them are non-separating cycles. In this case, $\exists$ sub-graph $\mathcal{G}'$ ($m_1, m_2 \notin \mathcal{G}'$) within $\mathcal{G}$ such that when some (not all) nodes in $\mathcal{G}'$ are used to construct vw -contained cycles (note that at least 2 nodes in $\mathcal{G}'$ are used), there exist some nodes that are disconnected to monitors in set A ($A \subset V(\mathcal{G}')$) no matter which of these constructed cycles is deleted. This case only happens when nodes in A are disconnected to monitors when v and w are deleted, contradicting

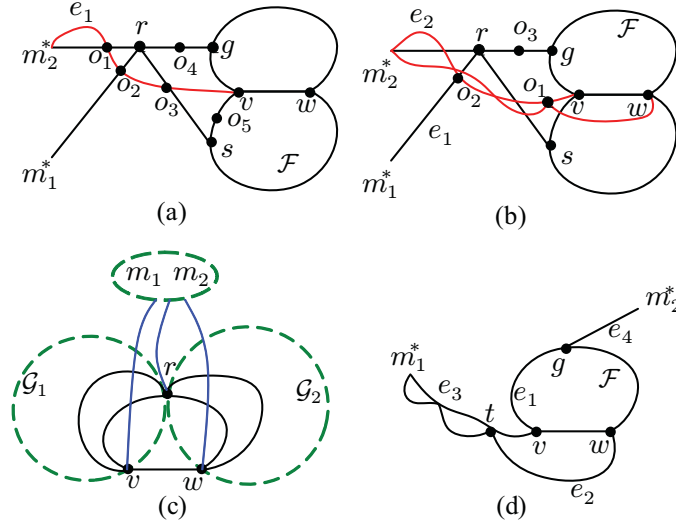


Figure E.4: Construction of two cycles and two paths.

Proposition D.2. Thus, there exists non-separating cycle $\mathcal{C}_1$.

3) Since cycles can be selected to ensure that $\mathcal{C}_1$ is a non-separating cycle; therefore, $\mathcal{C}_1$ is written as $\mathcal{F}$ ($\mathcal{F}$ denotes a non-separating cycle, shown in Mathematical Notations) instead in the sequel. Since $\mathcal{G}$ is connected, there exist simple path $\mathcal{P}_1$ connecting one monitor and a node on $\mathcal{F} - v - w$, and simple path $\mathcal{P}_2$ connecting the other monitor and a node on $\mathcal{C}_2 - v - w$ in $\mathcal{G}$. Note $\mathcal{P}_1$ (or $\mathcal{P}_2$) can be a degenerated path without containing any links. Properties of $\mathcal{F}$ and $\mathcal{C}_2$ are proved in 1) and 2). Now we prove $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ can be found.

(3.i). If all $\mathcal{P}_1$ ($m_1^* \in V(\mathcal{P}_1)$) must traverse m_2^* , then m_2^* is a cut-vertex in $\mathcal{G}$, contradicting Proposition D.2. Similarly, m_1^* is not an unavoidable node on $\mathcal{P}_2$ ($m_2^* \in V(\mathcal{P}_2)$).

(3.ii). With the aid of one end-point of vw , we prove that $\mathcal{C}_2$ and $\mathcal{P}_2$ can be re-selected when $\mathcal{P}_1$ and $\mathcal{P}_2$ have one unavoidable node for the given $\mathcal{F}$ and $\mathcal{C}_2$. Let $\mathcal{F} = vsw + vw$. For any $\mathcal{P}_1$ and $\mathcal{P}_2$, if they must have a common node, say r ($r \neq v$, $r \neq w$, and r can be equal to g), see Fig. E.4 (a)), then r cannot be a cut-vertex, because $\mathcal{G}$ is 2-vertex-connected. Therefore, there must be another path employing v or w (one and only one of v and w), say $m_2^*o_1 \cdots o_5vg$ ($r \notin V(m_2^*o_1 \cdots o_5vg)$); see

Mathematical Notations), to connect m_2^* and g . In addition, $m_2^*o_1 \cdots o_5vg$ might have common nodes $(o_1, \dots, o_5)$ with other paths. However, if o_4 or o_5 is the common node, then $\mathcal{P}_1$ and $\mathcal{P}_2$ do not need to traverse r to connect the two cycles. Thus, o_4 and o_5 are not common nodes. Therefore, $\mathcal{C}_2$ can be reselected, i.e., $\mathcal{C}_2 = \underline{vo_1ro_4gw} + vw$ with $\mathcal{P}_2 = m_2^*e_1o_1$ and $\mathcal{P}_1 = m_1^*rs$. Note $o_1\underline{ro_4g}$ is impossible to have unavoidable common nodes with $\mathcal{F}$; otherwise, m_2^* cannot connect to g when v and this unavoidable common node are deleted. Moreover, $\mathcal{P}_1$ might have common nodes, say o_2 or o_3 , with newly selected $\mathcal{C}_2$. This, however, does not affect the property that $|V(\mathcal{F}) \cap V(\mathcal{C}_2)| = 2$ or 3 , because $\overset{\circ}{v}o_3o_2o_1\underline{ro_4g}$ does not have common nodes with $\mathcal{F}$. Thus, the reselected $\mathcal{C}_2$ will not add any new common nodes between $\mathcal{F}$ and $\mathcal{C}_2$.

(3.iii). *With the aid of all end-points of vw , we prove that $\mathcal{C}_2$, $\mathcal{P}_1$ and $\mathcal{P}_2$ can be reselected when $\mathcal{P}_1$ and $\mathcal{P}_2$ have one unavoidable node r ($r \neq v$, $r \neq w$, and r can be equal to g) for the given $\mathcal{F}$ and $\mathcal{C}_2$.* Let $\mathcal{F} = \underline{vgw} + vw$ (see Fig. E.4 (b)). Suppose m_2^* can make use of both v and w , say $m_2^*o_1v$ and $m_2^*o_1w$, to connect nodes on $\overset{\circ}{v}g\overset{\circ}{w}$. We have $r, o_3 \notin V(m_2^*o_1v \cup m_2^*o_1w)$, since m_1^* and m_2^* must traverse r to connect to nodes on $\overset{\circ}{v}g\overset{\circ}{w}$ when v and w are not used. In addition, $m_2^*o_1v$ and $m_2^*o_1w$ do not have common nodes with $\mathcal{F} - v - w$. Therefore, utilizing $m_2^*o_1v$ and $m_2^*o_1w$, $\mathcal{C}_2$ and $\mathcal{P}_2$ can be reselected, i.e., if $\overset{\circ}{m_2^*o_1v}$ and $\overset{\circ}{m_2^*o_1w}$ have common nodes, say o_1 , then $\mathcal{C}_2 = \underline{vo_1w} + vw$ with $\mathcal{P}_2 = m_2^*e_2o_1$; otherwise, $\mathcal{C}_2 = \underline{ve_2m_2^*w} + vw$ with $\mathcal{P}_2 = \{m_2^*\}$. In the case that $\mathcal{P}_2 = m_2^*e_2o_1$, if $m_2^*e_2o_1$ has a common node with $m_1^*e_1r$, say o_2 , then based on previous operations, we can reselect $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$, i.e., $\mathcal{P}_1 = m_2^*ro_3g$ and $\mathcal{P}_2 = m_1^*e_1o_2o_1$. While for the unchanged $\mathcal{F}$, it does not have any common nodes with the reselected $\mathcal{C}_2$. Although $\mathcal{P}_1$ might have common nodes with the reselected $\mathcal{C}_2$, the property of $|V(\mathcal{F}) \cap V(\mathcal{C}_2)| = 2$ or 3 is undamaged.

(3.iv). *We prove the common node between $\mathcal{P}_1$ and $\mathcal{P}_2$ cannot be the common*

node between $\mathcal{F}$ and $\mathcal{C}_2$. According to (3.ii)–(3.iii), $\exists \mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{C}_2$ such that $\mathcal{P}_1$ and $\mathcal{P}_2$ do not have common node r . However, if r is the common node between $\mathcal{F}$ and $\mathcal{C}_2$ (see Fig. E.4 (c)), we can also prove it is impossible. In this case, as Fig. E.4 (c) shows, $V(\mathcal{G}_1) \cap V(\mathcal{G}_2) = \{r\}$ with $m_1, m_2 \notin V(\mathcal{G}_1)$ and $m_1, m_2 \notin V(\mathcal{G}_2)$ since m_1 and m_2 must use r or v or w to connect nodes on $\mathcal{F} - v - w - r$ and $\mathcal{C}_2 - v - w - r$. For the two cycles, we have $|\mathcal{G}_1 - r - v| \geq 1$ and $|\mathcal{G}_2 - r - w| \geq 1$ (since vw is the only common link between $\mathcal{F}$ and $\mathcal{C}_2$); therefore, nodes in $\mathcal{G}_1$ ($\mathcal{G}_2$) without monitors are separated when r and v (w) are deleted, contradicting Proposition D.2.

Based on (3.i)–(3.iv), therefore, $\exists \mathcal{P}_1$ and $\mathcal{P}_2$ without common nodes, i.e., $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$.

4) Now we prove that v and w are not unavoidable nodes on $\mathcal{P}_1$ and $\mathcal{P}_2$, i.e., $v, w \notin V(\mathcal{P}_1)$ and $v, w \notin V(\mathcal{P}_2)$. We first consider $\mathcal{P}_1$. In $\mathcal{G} - m_2^*$, if $\mathcal{P}_1$ must traverse an end-point of vw , say v , to connect m_1^* and a node on $\mathcal{F} - v - w$, then nodes on $\mathcal{F} - v - w$ are disconnected to m_1^* when v and m_2^* are deleted, contradicting Proposition D.2. Thus, it is impossible that $\mathcal{P}_1$ must traverse an end-point of vw . However, if $\mathcal{P}_1$ cannot avoid traversing one of v and w to connect m_1^* and $\mathcal{F} - v - w$, then two paths can be constructed. Let $\underline{ve_1gw} + vw$ be $\mathcal{F}$ (see Fig. E.4 (d)). The constructed two paths, connecting m_1^* and g , are $m_1^*e_3t\underline{ve_1g}$ and $m_1^*e_3t\underline{e_2wg}$ with $m_1^*e_3t\underline{v} \cap \overset{\circ}{\underline{ve_1g\underline{w}}} = \emptyset$ and $m_1^*e_3t\underline{e_2w} \cap \overset{\circ}{\underline{ve_1g\underline{w}}} = \emptyset$ (if they have intersections, $\mathcal{P}_1$ does not have to traverse v or w to connect to a node on $\overset{\circ}{\underline{ve_1g\underline{w}}}$). According to Proposition D.2, g must have a connection to m_2^* , $m_2^*e_4g$, with $m_2^*e_4g \cap m_1^*e_3t = \emptyset$ (if $m_2^*e_4g \cap m_1^*e_3t \neq \emptyset$, then $\mathcal{P}_1$ does not have to traverse v or w to connect to a node on $\overset{\circ}{\underline{ve_1g\underline{w}}}$). Therefore, $\mathcal{C}_2$ can be chosen as $\mathcal{C}_2 = \underline{vte_2w} + vw$ with $\mathcal{P}_2 = m_1^*e_3t$ and $\mathcal{P}_1 = m_2^*e_4g$ (if $\overset{\circ}{m_1^*e_3t\underline{v}}$ and $\overset{\circ}{m_1^*e_3t\underline{e_2w}}$ do not have common nodes, then $\mathcal{C}_2 = \underline{ve_3m_1^*e_2w} + vw$ with $\mathcal{P}_2 = \{m_1^*\}$). These two cycles and paths enable vw to be a cross-link identifiable via the method proposed in Section 3.2.2.1. When considering $\mathcal{P}_2$, the same argument (vw is identifiable via the method proposed in

Section 3.2.2.1) applies. Therefore, a cross-link can find $\mathcal{F}$, $\mathcal{C}_2$, $\mathcal{P}_1$ and $\mathcal{P}_2$ with all the properties in Lemma 3.4.

5) With the properties of (a) and (b) in Lemma 3.4. Let $\mathcal{P}$ be a path starting at one monitor and terminating at a node in $\mathcal{C} - v - w$ with $vw \in L(\mathcal{C})$ and $|V(\mathcal{P}) \cap V(\mathcal{C} - v - w)| > 1$. In this case, simply use the first common node between $\mathcal{P}$ and $\mathcal{C}$ as the termination of $\mathcal{P}$. Thus, $\mathcal{F}$, $\mathcal{C}_2$, $\mathcal{P}_1$ and $\mathcal{P}_2$ with properties (a), (b) and (c) in Lemma 3.4 can be found. $\square$

E.2 Proof of Lemma 3.5

E.2.1 Proof of Lemma 3.5-(a)

Using the method to calculate cross-link (see Section 3.2.2.1), all Case-A links in $\mathcal{H}$ can be identified. While for Case-B links, they can be further categorized into two classes: (i) Case B-1, $V(\mathcal{F} \cap \mathcal{C}_2) = \{v, w\}$ and all $\mathcal{P}_1$ must have a common node with $\mathcal{C}_2$, and (ii) Case B-2, $V(\mathcal{F} \cap \mathcal{C}_2) = \{v, w, r\}$, where r is another unavoidable common node (these three types of interior links are illustrated in Fig. 3.7 of Chapter 3).

Let vw be a Case-B link in $\mathcal{H}$ and $vw \in L(\mathcal{F})$. All other links on $\mathcal{F}$ can use the same non-separating cycle when constructing cycles and paths specified by Lemma 3.4, because nodes on $\mathcal{C}'_2$ of other links cannot be disconnected to monitors when $\mathcal{F}$ is deleted.

1) Let vw be a Case B-1 link.

(1.i). In Fig. E.5 (a), suppose xy is Case B-2 link on $\mathcal{F}$, then there is a common node s (there is at most one common node apart from x and y , proved in Lemma 3.4) on $\mathcal{F}$ and³ $\mathcal{C}_2^{xy}$. Since $\mathcal{F}$ is an induced graph, there must be a node, say v_1 , on $\underline{sv_1x}$ and a node, say v_2 , on $\underline{sv_2y}$. Note the meaning of common node s is that without

³Let $\mathcal{C}^l$ denote a cycle with link $l \in L(\mathcal{C}^l)$.

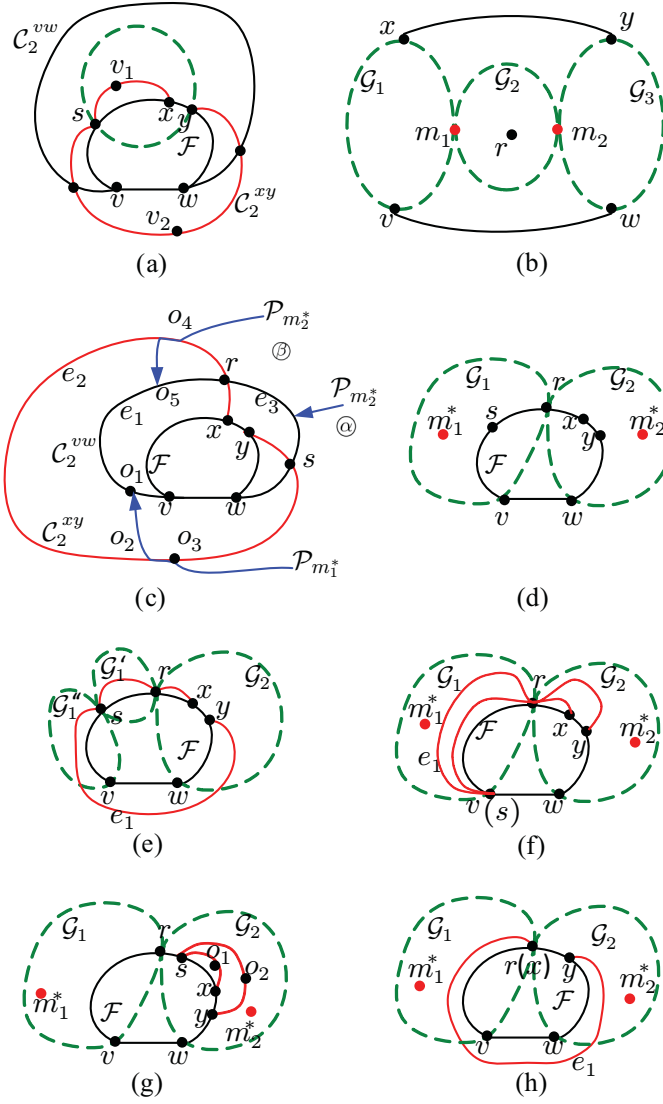


Figure E.5: Case-B links vw and xy cannot be in the same non-separating cycle.

using link xy , s is an unavoidable node to connect x and y . Thus, sv_1x cannot have common nodes with C_2^{vw} ; otherwise, s is not an unavoidable node. Meanwhile, if v_1 has a path to one monitor in $\mathcal{G} \setminus C_2^{vw}$, then x has a path to the same monitor in $\mathcal{G} \setminus C_2^{vw}$ as well, contradicting the assumption that vw is a Case B-1 link. Hence, for all paths connecting v_1 and monitors, they must traverse s or y . Thus, when s and y are deleted, v_1 is separated from m_1 and m_2 , contradicting Proposition D.2. This conclusion also holds when xy and s have common nodes with vw . As the position of s alters, however, the separated node might change. For instance, when $s = w$, v_2

is separated from m_1 and m_2 when x and w are deleted. Therefore, xy cannot be a Case B-2 link on non-separating cycle $\mathcal{F}$.

(1.ii). Suppose there is another Case B-1 link xy on $\mathcal{F}$ and both $\mathcal{C}_2^{vw}$ and $\mathcal{C}_2^{xy}$ must traverse m_1 and m_2 . Then graph $\mathcal{G}$ can be reorganized as Fig. E.5 (b), which is composed of sub-graphs $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$ and links vw , xy . There is at least one node, say r , in $\mathcal{G}_2 - m_1 - m_2$, because we have assumed direct link m_1m_2 does not exist in $\mathcal{G}$. Thus, the graph is disconnected when m_1 and m_2 are deleted, contradicting Proposition D.2. Therefore, it is impossible that $\mathcal{C}_2^{vw}$ and $\mathcal{C}_2^{xy}$ must traverse both m_1 and m_2 .

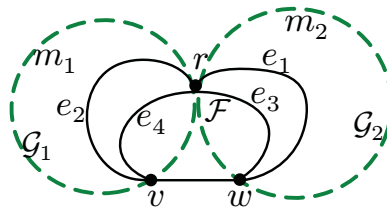
(1.iii). Since vw is a Case B-1 link, all possible $\mathcal{P}_1$ must intersect $\mathcal{C}_2^{vw}$. Thus, there exist path $\mathcal{P}_{m_1^*} := \mathcal{P}(m_1^*, v_1)$ and $\mathcal{P}_{m_2^*} := \mathcal{P}(m_2^*, v_2)$ with $v_1, v_2 \in V(\mathcal{C}_2^{vw})$ (v_1 and v_2 can be v and w) and $\mathcal{P}_{m_1^*} \cap \mathcal{P}_{m_2^*} = \emptyset$ (If $\mathcal{P}_{m_1^*} \cap \mathcal{P}_{m_2^*} \neq \emptyset$, the common node is a cut-vertex). Suppose there is another Case B-1 link xy on $\mathcal{F}$ (see Fig. E.5 (c)). Then the associated $\mathcal{C}_2^{xy}$ ($V(\mathcal{C}_2^{xy} \cap \mathcal{F}) = \{x, y\}$) must have two common nodes (since both vw and xy are Case B-1 links) with $\mathcal{C}_2^{vw}$, say r and s (we have proved that r and s cannot be both monitors in (1.ii)). Since xy is another Case B-1 link, if $\mathcal{P}_{m_1^*}$ connects to $\overset{\circ}{r}e_1\overset{\circ}{v}\overset{\circ}{w}\overset{\circ}{s}$, it must have common nodes with $\overset{\circ}{r}e_2\overset{\circ}{s}$, say the common node is o_3 (the number of common nodes maybe greater than one, say both o_2 and o_3). In addition, we have $o_3 \neq r \neq s$, since if o_3 must overlap with r or s , then it means v cannot connect to monitors when r and s are deleted, which is impossible. In Fig. E.5 (c), let o_1 be another node, which can be equal to v , on $\mathcal{C}_2^{vw}$. Now we consider the locations of $\mathcal{P}_{m_1^*}$ and $\mathcal{P}_{m_2^*}$. If $\mathcal{P}_{m_2^*}$ ends at re_3s (location @ in Fig. E.5 (c)), then $\mathcal{P}_{m_1^*}$ cannot end at $\overset{\circ}{r}e_1\overset{\circ}{v}\overset{\circ}{w}\overset{\circ}{s}$, because xy can select $xre_3sy + xy$ as $\mathcal{C}_2^{xy}$, and then path $m_1^*o_3o_2o_1v$ connecting m_1^* and v does not intersect with the newly selected $\mathcal{C}_2^{xy}$, resulting xy to be a non-Case B-1 link, contradicting the assumption that xy is a Case B-1 link. Therefore, $\mathcal{P}_{m_1^*}$ also ends at re_3s . In this case, however, v is disconnected to monitors when r and s (r and s cannot be both monitors according

to (1.ii)) are deleted, contradicting Proposition D.2. Now we change the location of $\mathcal{P}_{m_2^*}$. If no $\mathcal{P}_{m_1^*}$ and $\mathcal{P}_{m_2^*}$ end at $\underline{re_3s}$, then both $\mathcal{P}_{m_1^*}$ and $\mathcal{P}_{m_2^*}$ (location $\textcircled{\beta}$ in Fig. E.5 (c)) end at $\overset{\circ}{r}\underline{e_1vw}\overset{\circ}{s}$. In this case, $\mathcal{C}_2^{xy}$ can be reselected, i.e., $\mathcal{C}_2^{xy} = \underline{xre_3sy} + xy$ with $\mathcal{P}_2^{xy} = m_2^*o_4r$ and $\mathcal{P}_1^{xy} = m_1^*o_3o_2o_1v$. Thus, xy with $\mathcal{P}_1^{xy} \cap \mathcal{P}_2^{xy} = \emptyset$, which is a cross-link (see Section 3.2.2.1), is not a Case B-1 link, contradicting the assumption of xy being a Case B-1 link. This conclusion also holds when $y = w$ (or $x = v$). Thus, $\mathcal{F}$ with Case B-1 link vw cannot have another Case B-1 link.

2) Let vw be a Case B-2 link. For vw , suppose all cycles must traverse r , then $\mathcal{G}$ consists of sub-graph $\mathcal{G}_1$, $\mathcal{G}_2$ and link vw (see Fig. E.5 (d)). In addition, each of $\mathcal{G}_1$ and $\mathcal{G}_2$ has a monitor in it; otherwise, $\mathcal{G}_1$ ($\mathcal{G}_2$) is separated from monitors when r and v (w) are deleted, contradicting Proposition D.2.

(2.i). Suppose $xy \in L(\mathcal{G}_2)$ (see Fig. E.5 (e)) is a Case B-2 link on the same non-separating cycle $\mathcal{F}$, all $\mathcal{C}_2^{xy}$ must traverse a node, say s , on $\mathcal{F}$. If s is on $\overset{\circ}{v}\underline{sr}\overset{\circ}{r}$, $\mathcal{G}_1$ is further split into two sub-graphs ($\mathcal{G}'_1$ and $\mathcal{G}''_1$), contradicting the claim that $\mathcal{F} - x - y$ and $\mathcal{C}_2^{xy} - x - y$ cannot have two common nodes. Thus, s cannot be on $\overset{\circ}{v}\underline{sr}\overset{\circ}{r}$. If $s = v$, then path $\underline{xv_1y}$ is required. Since vw is a Case B-2 link, $\underline{xv_1y}$ must traverse r as well, resulting that $\mathcal{C}_2^{xy} - xy$ contains a cycle (see Fig. E.5 (f)), contradicting the basic requirement of simple paths. To avoid employing cycles, $\mathcal{C}_2^{xy}$ must be in $\mathcal{G}_2$ and s must be on $\underline{rxyw}$ (see Fig. E.5 (g)). Since $\mathcal{F}$ is induced, there exist nodes on $\overset{\circ}{s}o_1\overset{\circ}{x}$ and $\overset{\circ}{s}o_2\overset{\circ}{y}$. According to the properties of a non-separating cycle, o_1 and o_2 have connections to m_2^* in $\mathcal{G}_2 \setminus \mathcal{F}$ (in Lemma 3.4, we have proved the existence of $\mathcal{F}$). Using these connections (without containing s , x and y), $\mathcal{C}_2^{xy}$ without traversing s can be found, contradicting the assumption that xy is a Case B-2 link. When $x = r$ or $y = w$ or $xy \in L(\mathcal{G}_1)$, the same argument applies. Thus, xy cannot be a Case B-2 link on non-separating cycle $\mathcal{F}$ containing a Case B-2 link vw .

(2.ii). Suppose $xy \in L(\mathcal{G}_2)$ (see Fig. E.5 (h)) is a Case B-1 link on the same non-separating cycle $\mathcal{F}$, we have $r = x$ or $r = y$, since $\mathcal{F}$ and $\mathcal{C}_2^{xy}$ cannot have



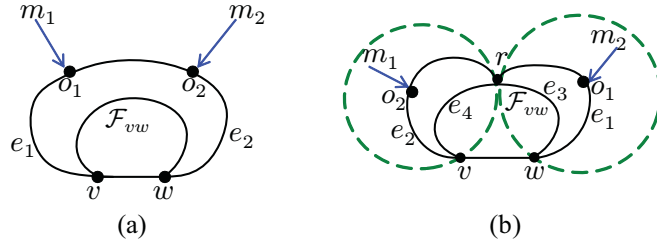
common nodes, apart from x and y . If $r = x$, there should be path re_1y and re_1y cannot have any links outside $\mathcal{G}_1$ and $\mathcal{G}_2$; therefore, $re_1y \subset \mathcal{G}_2$. In this case, there is a path $\mathcal{P}(m_1^*, v)$ ($r \notin V(\mathcal{P}(m_1^*, v))$). If r must be on $\mathcal{P}(m_1^*, v)$, then v is disconnected to monitors when r and w are deleted.) connecting m_1^* and v without intersecting re_1y , contradicting the assumption that xy is a Case B-1 link. The same conclusion can be obtained when $r = y$. Thus, xy cannot be a Case B-1 link on non-separating cycle $\mathcal{F}$ containing a Case B-2 link vw ..

9

Lemma 3.5-(b) is proved as follows.

If vw is Case B-1 link, then all paths connecting nodes on $\mathcal{F}_{vw} - v - w$ and monitors must intersect with $\mathcal{C}_2^{vw}$. Therefore, m_1 and m_2 cannot be on $\mathcal{F}_{vw}$.

If vw is Case B-2 link and all paths (besides direct link vw) connecting v and w must traverse a monitor, say m_1 , then it means $r = m_1$ in Fig. E.6. Thus, m_2 is in either $\mathcal{G}_1$ or $\mathcal{G}_2$ (each sub-graph has at least two links; otherwise, the single link becomes a bridge when vw is deleted). Suppose m_2 is in $\mathcal{G}_1$, then nodes in $\mathcal{G}_2 - r - w$ are disconnected to monitors when r ($r = m_1$) and v are deleted (see Fig. E.6), contradicting Proposition D.2. Then obviously, it is impossible that $\mathcal{F}$ must

Figure E.7: Paths construction for Case-B link vw .

traverse both m_1 and m_2 .

Now suppose either m_1 or m_2 must be on $\mathcal{F}_{vw}$. Without loss of generality, let $m_1 \in V(\mathcal{G}_1)$ and $m_2 \in V(\mathcal{G}_2)$ (see Fig. E.6). We have proved there exists a non-separating cycle, denoted by $\mathcal{F}$, for any link vw with $vw \in L(\mathcal{F})$ (shown in Fig. E.6). Since $\mathcal{F}$ is an induced graph, re_1w cannot be a single link. Thus, there is at least one node on $\overset{\circ}{r}e_1\overset{\circ}{w}$. In $\mathcal{G}_2$, when re_3w is deleted, nodes on $\overset{\circ}{r}e_1\overset{\circ}{w}$ should be able to connect to monitors (in this case, the monitor can only be m_2); otherwise, $ve_4re_3w + vw$ is not a non-separating cycle. Therefore, for a non-separating cycle containing Case B-2 link vw , m_2 cannot be on re_3w . Similarly, m_1 cannot be on re_4v . Therefore, as long as a non-separating cycle $\mathcal{F}$ containing Case B-2 link vw can be found (the existence of $\mathcal{F}$ has been proved in Lemma 3.4), $\mathcal{F}$ is a non-separating cycle without traversing m_1 or m_2 .

2) Finally we prove for the non-separating cycle $\mathcal{F}_{vw}$ selected by (b) in Lemma 3.5, there exist simple paths $\mathcal{P}(m_1, v)$ and $\mathcal{P}(m_2, w)$ with $\mathcal{P}(m_1, v) \cap \mathcal{P}(m_2, w) = \emptyset$, $\mathcal{P}(m_1, v) \overset{\circ}{v} \cap \mathcal{F}_{vw} = \emptyset$ and $\mathcal{P}(m_2, w) \overset{\circ}{w} \cap \mathcal{F}_{vw} = \emptyset$.

(2.i). Suppose vw is Case B-1 link. Then there exist simple paths $\mathcal{P}(m_1, o_1)$ and $\mathcal{P}(m_2, o_2)$, where $o_1, o_2 \in V(\mathcal{C}_2^{vw})$ (see Fig. E.7 (a)). $\mathcal{P}(m_1, o_1)$ and $\mathcal{P}(m_2, o_2)$ do not have unavoidable common node; otherwise, monitors cannot connect to nodes on $\mathcal{C}_2^{vw}$ when the unavoidable common node is deleted. For Case-B link vw , we have proved the existence of $\mathcal{F}_{vw}$ with $m_1, m_2 \notin V(\mathcal{F}_{vw})$. Therefore, simple paths $\mathcal{P}(m_1^*, v)$ and $\mathcal{P}(m_2^*, w)$ with $\mathcal{P}(m_1^*, v) \cap \mathcal{P}(m_2^*, w) = \emptyset$, $\mathcal{P}(m_1^*, v) \overset{\circ}{v} \cap \mathcal{F}_{vw} = \emptyset$ and

$\mathcal{P}(m_2^*, w) \overset{\circ}{w} \cap \mathcal{F}_{vw} = \emptyset$ can be selected, i.e., $\mathcal{P}(m_1^*, v) = m_1 \underline{o_1 e_1} v$ and $\mathcal{P}(m_2^*, w) = m_2 \underline{o_2 e_2} w$.

(2.ii). Suppose vw is Case B-2 link. In Section E.2.2-1), we have proved the existence of non-separating cycle $\mathcal{F}_{vw}$ with $vw \in L(\mathcal{F}_{vw})$. In Fig. E.7 (b), let $F_{vw} = v \underline{e_4 r e_3} w + vw$. Without loss of generality, let m_1 be in $\mathcal{G}_1$ and m_2 be in $\mathcal{G}_2$. Then when re_3w is deleted, there must be a path connecting m_2 and a node on $\overset{\circ}{r e_1} \overset{\circ}{w}$, say $\mathcal{P}(m_2, o_1)$, where $o_1 \in \overset{\circ}{r e_1} \overset{\circ}{w}$. Similarly, there exists path $\mathcal{P}(m_1, o_2)$, where $o_2 \in \overset{\circ}{v e_2} \overset{\circ}{r}$. Therefore, simple paths $\mathcal{P}(m_1^*, v)$ and $\mathcal{P}(m_2^*, w)$ with $\mathcal{P}(m_1^*, v) \cap \mathcal{P}(m_2^*, w) = \emptyset$, $\mathcal{P}(m_1^*, v) \overset{\circ}{v} \cap \mathcal{F}_{vw} = \emptyset$ and $\mathcal{P}(m_2^*, w) \overset{\circ}{w} \cap \mathcal{F}_{vw} = \emptyset$ can be selected, i.e., $\mathcal{P}(m_1^*, v) = m_1 \underline{o_2 e_2} v$ and $\mathcal{P}(m_2^*, w) = m_2 \underline{o_1 e_1} w$. $\square$

E.3 Proof of Theorem 4.2

The three independent spanning trees $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ are constructed by the method in Section B, where $m_i = m_2$ when constructing $\mathcal{G}_{ex}^*$. Leveraging the same method used in the proof of Lemma D.3 (Section D.2.3), again, we color the links on $\mathcal{T}_1$ as blue, $\mathcal{T}_2$ as green, and $\mathcal{T}_3$ as red. In addition, each link in $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ is assigned a direction toward r (see Appendix B). In Lemma D.3 of Appendix D, we proved that cycle set C contains $||\mathcal{G}_m||$ linearly independent cycles, where $\mathcal{G}_m = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$. For any cycle in C , removing all involved virtual links and the resulting isolated nodes, we can obtain a monitor-to-monitor simple path since each cycle in C is obtained

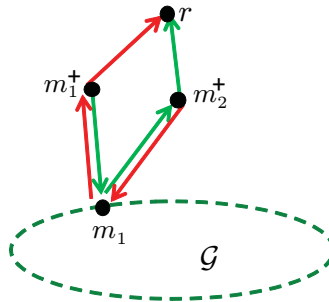


Figure E.8: Combining red and green paths w.r.t. m_1 .

by combining two internally vertex disjoint paths ($\mathcal{X}^{(i)}$ and $\mathcal{X}^{(j)}$) ($i, j \in \{1, 2, 3\}$, $i \neq j$) along two independent spanning trees and virtual links only appear at the end of $\mathcal{X}^{(i)}$ and $\mathcal{X}^{(j)}$. Therefore, following the same operation to each cycle in C iteratively, we can generate a new set Y with each element $\mathcal{Y}_i$ representing a monitor-to-monitor simple paths. In this proof, we first investigate on the number of linearly independent paths in Y . Note that some paths in Y might contain more than 2 monitors. Hence, we then show how to shorten these paths in the next section.

E.3.1 Number of Linearly Independent Paths After Removing All Virtual Links

Let $\mathbb{V}$ denote the set of virtual links⁴ in $\mathcal{G}_m$. The goal is to prove the number of linearly independent paths in Y is $||\mathcal{G}_m|| - |\mathbb{V}|$, and these linearly independent paths are sufficient to identify all tree links in the original graph $\mathcal{G}$.

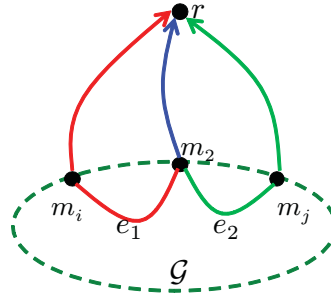
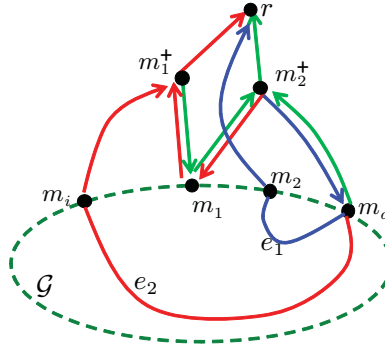
For all redundant paths generated by mapping from C to Y , they can be divided into 3 categories, i.e., 1) trivial topology, 2) linearly dependent paths, and 3) duplicate paths. Note in Fig. E.8–Fig. E.14, for all paths (or path segments) colored as blue/green/red, they represent virtual links if they are outside $\mathcal{G}$; and real simple paths otherwise.

1) Trivial topology (a graph with no nodes or links). As shown in Fig. E.8, combining the red and green paths w.r.t. m_1 ($\mathcal{G}_m$ involves $rm_1^+m_1m_2^+r$), cycle $rm_1^+m_1m_2^+r$ is formed. However, the corresponding path of $rm_1^+m_1m_2^+r$ after removing the virtual links and the resulting isolated nodes is empty. Therefore, this trivial path has no contribution for identifying real links in $\mathcal{G}$.

2) Linearly dependent paths. The following 3 cases can generate redundant linearly dependent paths.

(2.i). Consider the three cycles constructed w.r.t. m_2 (shown in Fig. E.9). The

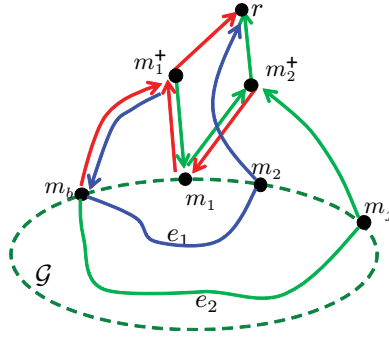
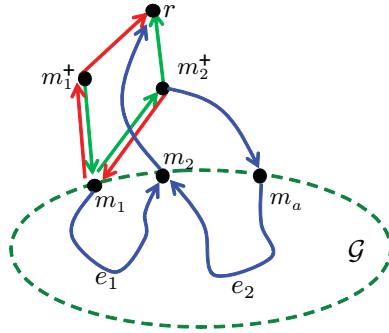
⁴ $\mathcal{G}_m$ is a spanning graph of $\mathcal{G}_{ex}^*$. Thus, the number of virtual links in $\mathcal{G}_m$ is less than that in $\mathcal{G}_{ex}^*$.

Figure E.9: Path construction w.r.t. m_2 .Figure E.10: Path construction w.r.t. m_a .

paths associated with *blue + red* cycle and *blue + green* cycle are $m_i e_1 m_2$ and $m_2 e_2 m_j$, respectively. However, the path associated *red+green* cycle, $m_i e_1 m_2 e_2 m_j$, is the sum of previously constructed paths $m_i e_1 m_2$ and $m_2 e_2 m_j$. Therefore, redundant path $m_i e_1 m_2 e_2 m_j$ is linearly dependent with the other two.

(2.ii). Now consider the case shown in Fig. E.10. For $\mathcal{G}_m$, there exists one and only one monitor m_a with link $m_2^+ m_a$ colored as blue in the $m_2^+ \rightarrow m_a$ direction, according to the rules for blue tree construction. Now consider m_a . According to the s-t numbering rule and the processing of ear decomposition, we have $f(m_2^+) < f(m_a)$ and $g(m_2^+) < g(m_a)$; therefore, $m_2^+ m_a$ is colored⁵ as green in the other direction. Then $m_2 e_1 m_a$ and $m_i e_2 m_a$ are two paths obtained from *green + blue* cycle and *green + red* cycle w.r.t. m_a , respectively. However, the path generated by the *blue + red* cycle is the sum of $m_2 e_1 m_a$ and $m_i e_2 m_a$, thus redundant in Y .

⁵Note m_a might have more than one neighbor w with $f(w) < f(m_a)$ and $g(w) < g(m_a)$, in which case we can still choose $m_2^+ m_a$ to color it as green.

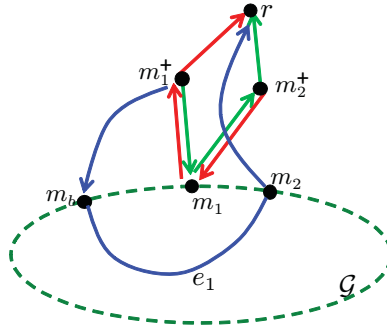
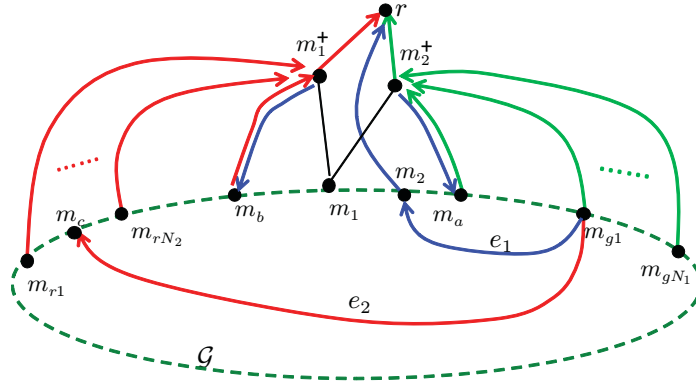
Figure E.11: Path construction w.r.t. m_b .Figure E.12: Duplicate paths w.r.t. m_1 and m_2^+ .

(2.iii). Analogously, there exists one and only one monitor m_b with link $m_1^+m_b$ colored as blue in the $m_1^+ \rightarrow m_b$ direction (see Fig. E.11). Based on the non-separating ear decomposition, we have $f(m_1^+) > f(m_b)$ and $g(m_1^+) < g(m_b)$; therefore, $m_1^+m_b$ is colored⁶ as red in the other direction. Then $m_be_1m_2$ and $m_be_2m_j$ are two paths obtained from *red* + *blue* cycle and *red* + *green* cycle w.r.t. m_b , respectively. However, the path generated by the *blue* + *green* cycle is the sum of $m_be_1m_2$ and $m_be_2m_j$, thus not providing new information for link identifications in $\mathcal{G}$.

3) Duplicate paths. Each of the following 3 cases can generate the same path twice.

(3.i). We have considered combining the red and green paths w.r.t. m_1 . Now consider the paths associated with the *blue* + *red* cycle and *blue* + *green* cycle. As Fig. E.12 displays, the paths associated with these two cycles are exactly the same,

⁶Note m_b might have more than one neighbor u with $f(u) > f(m_a)$ and $g(u) < g(m_a)$, in which case we can still choose $m_1^+m_b$ to color it as red.

Figure E.13: Duplicate paths w.r.t. m_1^+ .Figure E.14: Other virtual links connecting to m_1^+ or m_2^+ .

i.e., path $m_1e_1m_2$ is generated twice.

(3.ii). Following the similar argument, the paths associated with the *blue + red* cycle and *blue + green* cycle w.r.t. m_2^+ are also the same, i.e., path $m_2e_2m_a$ (shown in Fig. E.12) is generated twice when mapping from C to Y . Note m_a cannot be the same as m_2 since m_2 only appears in the last ear. Therefore, path $m_2e_2m_a$ is non-trivial.

(3.iii). Similar to (3.i) and (3.ii), the paths associated with the *blue + red* cycle and *blue + green* cycle w.r.t. m_1^+ are also the same, i.e., $m_be_1m_2$ as shown in Fig. E.13 appears twice when mapping from C to Y . In addition, $m_be_1m_2$ is non-trivial since m_2 has the highest ear level which means $m_b \neq m_2$.

4) We have discussed 7 cases in 1)–3), each case providing a redundant path when mapping from C to Y . For the spanning graph $\mathcal{G}_m$, the minimum number of

virtual links in $\mathcal{G}_m$ is also 7, i.e., rm_1^+ , rm_2^+ , $m_1^+m_1$, $m_2^+m_1$, $m_2^+m_a$ (Fig. E.10), and $m_1^+m_b$ (Fig. E.11). However, in addition to these 7 virtual links, there might exist other virtual links in $\mathcal{G}_m$. As Fig. E.14 shows, suppose there are other N_1 virtual links (colored as green and no color on the other direction as $m_1^+m_b$ has been colored as blue) connecting m_2^+ and m_{gi} ($i = 1, \dots, N_1$) and N_2 virtual links (colored as red and no color on the other direction as $m_1^+m_b$ has been colored as blue) connecting m_1^+ and m_{ri} ($i = 1, \dots, N_2$). For the 3 paths constructed w.r.t. each node in $\{m_{g1}, \dots, m_{gN_1}, m_{r1}, \dots, m_{rN_2}\}$, there exists one path which is linearly dependent with the other two. To prove this claim, consider m_{g1} as an example. Two paths⁷ $m_2e_1m_{g1}$ and $m_ce_2m_{g1}$ (see Fig. E.14) can be obtained from the *green* + *blue* cycle and *green* + *red* cycle w.r.t. m_{g1} . However, the path associated with the *blue* + *red* cycle w.r.t. m_{g1} is the sum of $m_2e_1m_{g1}$ and $m_ce_2m_{g1}$, thus linearly dependent with the other two. The same argument can be applied to other nodes in $\{m_{g1}, \dots, m_{gN_1}, m_{r1}, \dots, m_{rN_2}\}$. Therefore, we can identify another $N_1 + N_2$ linearly dependent paths in Y regarding these $N_1 + N_2$ virtual links.

In sum, 1)–4) cover all the possible cases of redundant paths when mapping from C to Y and the number of these redundant paths is $7 + N_1 + N_2$, which is also the number of virtual links (denoted by $|\mathbb{V}|$) in $\mathcal{G}_m$. Therefore, removing these redundant paths, the cardinality of the resulting path set Y' is $||\mathcal{G}_m|| - (7 + N_1 + N_2) = ||\mathcal{G}_m|| - |\mathbb{V}|$.

Now we can explore if Y' is sufficient to identify all tree links in $\mathcal{G}_{ex}^*$. Recall that each path (for identifying tree links) constructed by STPC consists of one or two path segments, each path segment terminating at the first monitor it encounters. If we let each segment terminate at the last monitor before traversing a virtual link and combine any two segments w.r.t. real node v (v can be either monitor or non-monitor), then a new measurement path set J is formed. It is easy to show (using

⁷Note m_c can be any node in $\{m_{r1}, \dots, m_{rN_2}, m_b\}$ in Fig. E.14.

STLI) that this new path set J is sufficient to identify all tree links in $\mathcal{G}_{ex}^*$. Recall the mapping process from cycle set C to path set Y . Each path in Y is obtained by removing the virtual links and all resulting isolated nodes of the corresponding cycle in C . Then each path $\mathcal{P}_J$ in J can be obtained by applying the same operation to the corresponding cycle C_J ($C_J \supset \mathcal{P}_J$) in C ; therefore, each path in J must be involved as one element in Y . Accordingly, Y is sufficient to identify all tree links in $\mathcal{G}_{ex}^*$. Since Y' is obtained by removing redundant paths in the procedure of mapping C to Y , then Y' is also sufficient to identify all tree links in $\mathcal{G}_{ex}^*$, i.e., the measurement matrices associated with J , Y and Y' have the same column rank.

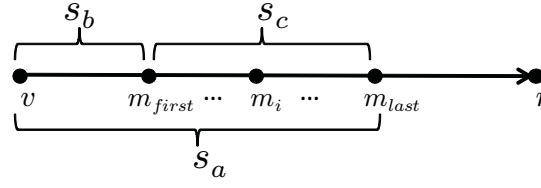
In sum, for graph $\mathcal{G}_m$ with $||\mathcal{G}_m|| - |\mathbb{V}|$ real links, the property of Y' is that it contains exactly $||\mathcal{G}_m|| - |\mathbb{V}|$ paths, with each path containing only tree links in $\mathcal{G}_{ex}^*$. Therefore, the measurement matrix associated with Y' is a square matrix with full rank.

E.3.2 Number of Distinct Paths by STPC

we now prove that all paths with more than 2 monitors in Y' can be shortened to form a new path set Z which can cover all the paths obtained by STPC (Algorithm 4.1) for tree link identification, i.e., paths obtained by line 1–11 in STPC algorithm.

1) First, we consider the number of generated paths w.r.t. to node v (line 5–15 in STPC).

(1.i). If v is a monitor in $\{m_2, m_a, m_b, m_{g1}, \dots, m_{gN_1}, m_{r1}, \dots, m_{rN_2}\}$ (see Fig. E.14), then one path in $\{\mathcal{P}_{v1}, \mathcal{P}_{v2}, \mathcal{P}_{v3}\}$ contains no links, resulting to be an invalid path and discarded without appending to $\mathbf{R}$ in line 10 of STPC. In this case, only two paths are generated in line 6. In 2) and 4) of Section E.3.1, we have shown that one linearly dependent path, which is the combination of the other two paths w.r.t. the same node, has been removed, thus not existed in Y' . Therefore, in Y' , there are also two paths w.r.t. to nodes belonging to $\{m_2, m_a, m_b, m_{g1}, \dots, m_{gN_1}, m_{r1},$

Figure E.15: Shortening a long path in Y' .

$\dots, m_{rN_2}\}$.

(1.ii). If v is m_1 , then line 6 of STPC only generates one path, since the other two paths contain only one node, i.e., m_1 itself. In this case, we have also shown that only path $(m_1e_1m_2)$ colored as blue in Fig. E.12) w.r.t. m_1 is retained in Y' after removing one invalid (Section E.3.1-1)) and one duplicate (Section E.3.1-3)-(3.i)) path.

(1.iii). Apart from the above two cases, each node in $\mathcal{G}$ corresponds to 3 measurement paths both in Y' and $\mathcal{P}_{STPC}$, where $\mathcal{P}_{STPC}$ is the path set by STPC after executing line 1–11.

2) Second, we compare the differences between the two path segments obtained by the two methods w.r.t. to node v on the same tree $\mathcal{T}_i$. Let $\mathcal{S}_a$ and $\mathcal{S}_b$ denote the corresponding path segments associated with Y' and $\mathcal{P}_{STPC}$, respectively. Both $\mathcal{S}_a$ and $\mathcal{S}_b$ start at v and go toward r along the same tree $\mathcal{T}_i$. The only difference is that $\mathcal{S}_a$ terminates at the last monitor before traversing a virtual link while $\mathcal{S}_b$ terminates at the first monitor it encounters (as shown in Fig. E.15). In Fig. E.15, observe that w.r.t. m_{first} , $\mathcal{S}_c$ is constructed and $W_{\mathcal{S}_c}$ can be computed through measuring the paths constructed w.r.t. m_{first} in Y' . Therefore, when constructing path segment $\mathcal{S}_a$ w.r.t. v , there is no need to terminate at m_{last} . $\mathcal{S}_a$ can also terminate at m_{first} since $W_{\mathcal{S}_c}$ can be known from other path measurements directly (there are only two monitors on $\mathcal{S}_c$) or indirectly (in Fig. E.15, $W_{m_i \rightarrow m_{last}}$ and $W_{m_{first} \rightarrow m_i}$ are obtained from paths constructed w.r.t. m_i and m_{first} , respectively). Using this operation, each path with more than 2 monitors in Y' can be shortened.

3) Let Y'' denote the path set obtained from Y' after the above shortening process. We can show in set Y'' , for a monitor m^* which is not in $\{m_2, m_a, m_b, m_{g1}, \dots, m_{gN_1}, m_{r1}, \dots, m_{rN_2}\}$, the corresponding measurement paths can be further shortened. Let $\mathcal{S}_1^*, \mathcal{S}_2^*$ and $\mathcal{S}_3^*$ denote the corresponding shortened path segments w.r.t. m^* , then the associated measurement paths in Y'' are $\mathcal{S}_1^* \cup \mathcal{S}_2^*, \mathcal{S}_2^* \cup \mathcal{S}_3^*$ and $\mathcal{S}_3^* \cup \mathcal{S}_1^*$, which can be further shortened to $\mathcal{S}_1^*, \mathcal{S}_2^*$ and $\mathcal{S}_3^*$, respectively, forming the same 3 paths as those obtained by STPC. Suppose the newly formed set from Y'' by path shortening is Z , then Z is also sufficient to identify all tree links in $\mathcal{G}_m$. This is because, from the perspective of linear algebra, this shortening operation means that one part of the original linear equation can be eliminated by subtracting the linear combinations of some other rows; therefore, the resulting matrix rank is the same as the original matrix. Since we have proved that the measurement matrix associated with Y' has a full rank, the measurement matrix associated with Z also has a full rank.

To identify tree links in $\mathcal{G}_{ex}^*$, based on the above three arguments, we know that the number of paths constructed w.r.t. v are the same in both Y' and $\mathcal{P}_{STPC}$. Moreover, w.r.t. the same node and along the same independent spanning tree(s), the paths obtained by these two methods are exactly the same after the path shortening operation. Thus, the newly generated path set Z can cover all the paths selected by STPC for tree link identification. The number of paths in Z is the number of tree links in $\mathcal{G}_{ex}^*$; therefore, the number of distinct paths constructed by line 1–11 in STPC is exactly the number of tree links in $\mathcal{G}_{ex}^*$.

Finally, we consider the non-tree links of the original graph $\mathcal{G}$. As $\mathcal{G}_m = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$, all non-tree links are involved in set $L(\mathcal{G} \setminus \mathcal{G}_m)$. With the knowledge of tree link metrics, based on the auxiliary algorithm of STPC for non-tree link identification, in line 12–15, each non-tree link l (with link metric W_l) corresponds to one new path measurement involving W_l as the only unknown link metric. Therefore, the number of newly added paths by line 12–15 equals the number of non-tree links in $\mathcal{G}$ and

they are linearly independent with all other selected paths. A link in $\mathcal{G}$ is either a tree or non-tree link; therefore, the number of distinct paths constructed by STPC equals the number of links in $\mathcal{G}$. $\square$

E.4 Proof of Theorem 5.3

In Theorem 3.3 and 3.7, the prerequisite for network identifiability is that all involved links can be used for constructing measurement paths. In DAIL, we sequentially consider each triconnected component which possibly contains virtual links. These virtual links, however, do not exist in real networks. To tackle with this issue, Claim 1 (see Appendix D) states that a replacement path can be found for a virtual link whose end-points $\{v_1, v_2\}$ form a (or an equivalent) Type-0-VC w.r.t. $\mathcal{T}$. We will discuss the virtual links, not covered by Claim 1, in the following link identifications for the three categories (where there are four sub-categories for Category 2; see Appendix A for the definition of the four sub-categories) of triconnected components.

1) Category 1.

It suffices to only consider the case of a triconnected component $\mathcal{T}$ with 4 vantages. This is because, for a triconnected component $\mathcal{T}$ with more than 4 vantages, only four of them are useful for identifying $\mathcal{T}$. Since $|V(\mathcal{T})| \geq 4$, $\mathcal{T}$ must be 3-vertex-connected, i.e., neither a triangle nor a bond. There are 3 possible scenarios after replacing all possible virtual links (according to Claim 1) in $\mathcal{T}$ by the corresponding real paths in neighboring components. (i) For the four vantages, each vantage-to-agent path is vertex disjoint with all others: As illustrated in Fig. E.16 (a), there exist 4 vertex disjoint vantage-to-agent paths $\mathcal{P}_1, \dots, \mathcal{P}_4$. Therefore, abstracting $\mathcal{P}_1, \dots, \mathcal{P}_4$ as single links, the graph in Fig. E.16 (a) satisfies the condition in Theorem 3.7, and thus $\mathcal{T}$ is fully identifiable. (ii) For two vantages in $\{\mu_1, \dots, \mu_4\}$,

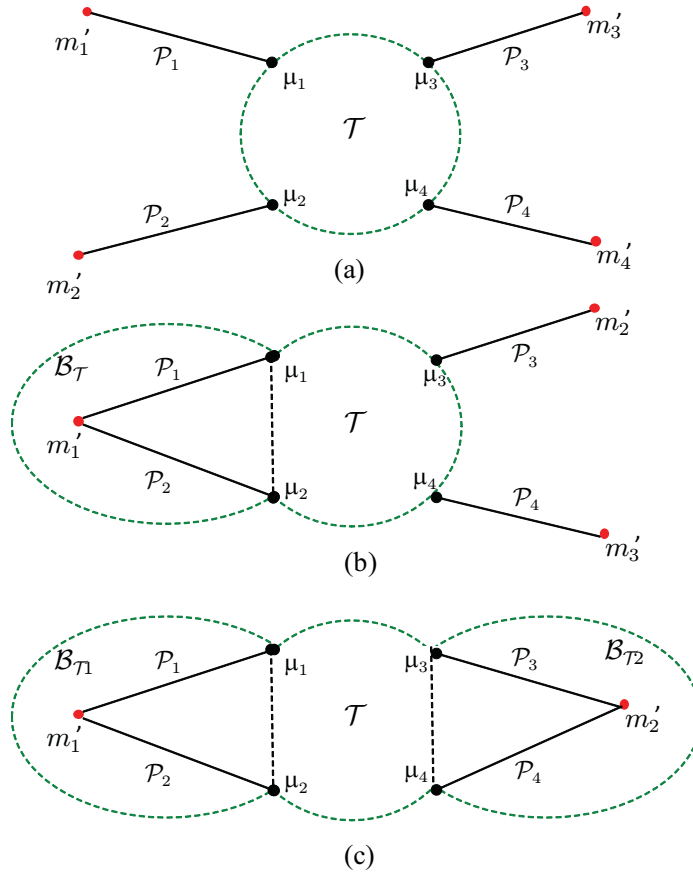


Figure E.16: Link identifications of Category 1.

say μ_1 and μ_2 , their vantage-to-agent paths are within the same biconnected component, as shown in Fig. E.16 (b). In this case, there exists at least one agent (m'_1) in the neighboring biconnected component $\mathcal{B}_T$, and two internally vertex disjoint paths $\mathcal{P}_1$ and $\mathcal{P}_2$ in Fig. E.16 (b). Hence, abstracting $\mathcal{P}_1, \dots, \mathcal{P}_4$ in Fig. E.16 (b) as single links, the graph in Fig. E.16 (b) also satisfies the condition in Theorem 3.7 even if $\mu_1\mu_2$ is a virtual link, and thus $\mathcal{T}$ is fully identifiable. (iii) For two pairs of vantages in $\{\mu_1, \dots, \mu_4\}$, the vantage-to-agent paths for each pair are within the same biconnected component, as shown in Fig. E.16 (c). Following the similar argument in (ii), the graph in Fig. E.16 (c) satisfies the condition in Theorem 3.7 even if $\mu_1\mu_2$ and $\mu_3\mu_4$ are virtual links as $\mathcal{T}$ is 3-vertex-connected, and thus $\mathcal{T}$ is fully identifiable. On top of Fig. E.16 (c), there may exist other pairs of vantages, whose vantage-to-agent paths are within the same biconnected component. In this case, these vantage-to-

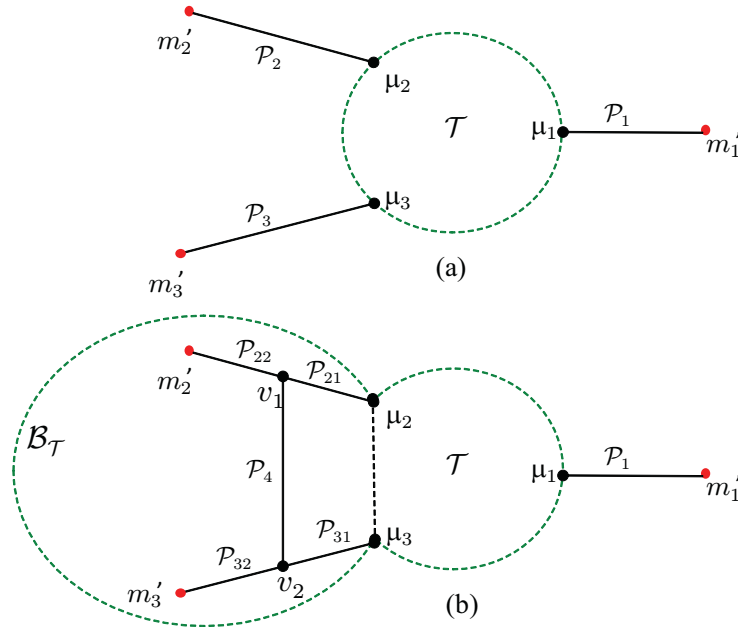


Figure E.17: Link identifications of Category 2.1.

agent paths can be ignored, since $\mathcal{P}_1, \dots, \mathcal{P}_4$ in Fig. E.16 (c) are already sufficient to identify $\mathcal{T}$.

2) Category 2.1.

Suppose $\mathcal{T}$ of Category 2.1 (Fig. E.17) is 3-vertex-connected, and all possible virtual links in $\mathcal{T}$ are replaced by the corresponding real paths in neighboring components of $\mathcal{T}$. There are two possible scenarios: (i) For the three vantages, each vantage-to-agent path is within a different neighboring biconnected component (all these neighboring biconnected components are within the same parent biconnected component): As illustrated in Fig. E.17 (a), there exist 3 vertex disjoint vantage-to-agent paths $\mathcal{P}_1, \dots, \mathcal{P}_3$. Therefore, abstracting $\mathcal{P}_1, \dots, \mathcal{P}_3$ as single links, the graph in Fig. E.17 (a) satisfies the condition in Theorem 3.7, and thus $\mathcal{T}$ is fully identifiable. (ii) For two vantages in $\{\mu_1, \dots, \mu_3\}$, say μ_2 and μ_3 , their vantage-to-agent paths are within the same neighboring biconnected component $\mathcal{B}_{\mathcal{T}}$, as shown in Fig. E.17 (b). In this case, there exist path $\mathcal{P}_4$ connecting v_1 and v_2 , and v_1 (v_2) has at least 3 neighbors in $\mathcal{B}_{\mathcal{T}}$. Moreover, μ_2 and μ_3 must have at least two neighbors in $\mathcal{T}$ (we have assumed all other virtual links in $\mathcal{T}$ have been replaced by neighboring compo-

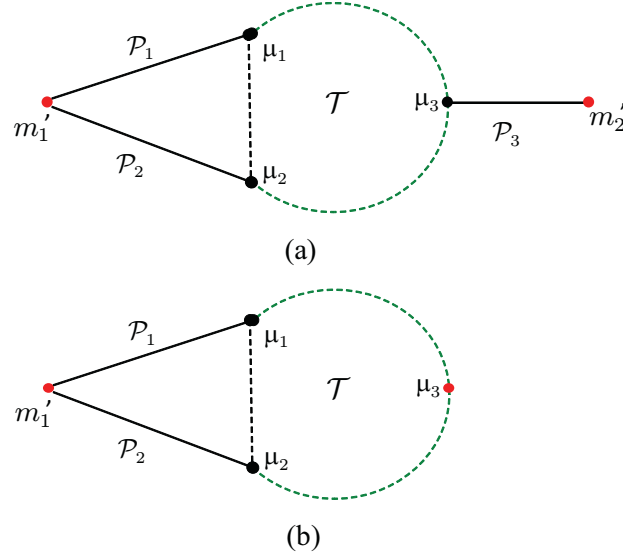


Figure E.18: Link identifications of Category 2.2.

nents), since $\mathcal{T}$ is 3-vertex-connected. Therefore, μ_2 and μ_3 are not nodes with only two neighbors, and thus they do not have to be monitors for complete network identification of Fig. E.17 (b) (see the explanation of MMP in Chapter 6). Abstracting $\mathcal{P}_1, \mathcal{P}_{21}, \mathcal{P}_{22}, \mathcal{P}_{31}, \mathcal{P}_{32}, \mathcal{P}_4$ in Fig. E.17 (b) as single links, the graph in Fig. E.17 (b) satisfies the condition in Theorem 3.7 even if $\mu_2\mu_3$ is a virtual link, and thus 3-vertex-connected component $\mathcal{T}$ is fully identifiable. Meanwhile, same argument applies if $\{\mu_1, \mu_2\}$ and/or $\{\mu_1, \mu_3\}$ are type- k -VCs ($k \geq 2$).

If $\mathcal{T}$ is a triangle, then the identifiability of $\mathcal{T}$ is determined by Lemma D.5.

3) Category 2.2.

Fig. E.18 illustrates the case of Category 2.2, where μ_1, μ_2 and μ_3 are the vantages in $\mathcal{T}$ and $\{\mu_1, \mu_2\}$ is a conjugate pair (see Definition 14 in Appendix A). Since the parent biconnected component of $\mathcal{T}$ is 2-vertex-connected, there exist two paths $\mathcal{P}_1$ and $\mathcal{P}_2$ connecting to an agent m'_1 . $\mathcal{P}_1$ and $\mathcal{P}_2$ are internally vertex disjoint, since if $\mathcal{P}_1$ and $\mathcal{P}_2$ must have a common node (except m'_1), then this common node is a cut-vertex, contradicting the property of the parent biconnected component being 2-vertex-connected (see Lemma D.4). Moreover, there exists path $\mathcal{P}_3$ connecting μ_3 and m'_2 with $\mathcal{P}_1 \cap \mathcal{P}_3 = \emptyset$ and $\mathcal{P}_2 \cap \mathcal{P}_3 = \emptyset$. By Claim 1, all virtual links in $\mathcal{T}$ (except

$\mu_1\mu_2$) can be replaced by the corresponding real paths in neighboring components; therefore, we only need to consider one virtual link $\mu_1\mu_2$ (if any) in Fig. E.18. In the proof of Lemma 5.1, Fig. 5.5 illustrates that, using two monitors, if one monitor has only one neighbor v , then all neighboring links of v are unidentifiable. Therefore, in the case of employing two monitors, this neighboring node of each monitor v is an effective monitor. We can therefore abstract m'_1 and μ_3 as two monitors (by Lemma 5.1) and $\mathcal{P}_1$ and $\mathcal{P}_2$ as single links in Fig. E.18 (a) to obtain Fig. E.18 (b). Fig. E.18 (b) satisfies the interior graph identifiability conditions (Theorem 3.3) and exterior link unidentifiability conditions (Corollary 3.2) naturally if there exists real link $\mu_1\mu_2$. Now we consider that there is no link $\mu_1\mu_2$ in the original graph, i.e., $\mu_1\mu_2$ is a virtual link. If $\mathcal{T}$ is a triangle, then the two exterior links ($\mu_1\mu_3$ and $\mu_2\mu_3$ if any) in Fig. E.18 (b) are unidentifiable according to Corollary 3.2. Now consider the case that $\mathcal{T}$ is 3-vertex-connected. Deleting any two links⁸ in $\mathcal{T}$, the resulting graph is connected as 3-vertex-connectivity of $\mathcal{T}$ implies 3-edge-connectivity. Deleting $\mathcal{P}_1$ (or $\mathcal{P}_2$) and one link in $\mathcal{T}$, we also get a connected remaining graph. Thus, Fig. E.18 (b) satisfies Condition ① in Theorem 3.3. Now consider deleting some vertices in $\mathcal{T}$. Deleting any two vertices in $\mathcal{T}$, the remaining graph of $\mathcal{T}$ is still connected as $\mathcal{T}$ is 3-vertex-connected. In this case, if m'_1 is isolated (μ_1 and μ_2 are deleted), m'_1 can reconnect to the remaining part of $\mathcal{T}$ by added link $m'_1\mu_3$ (see Condition ② in Theorem 3.3). When deleting m'_1 and a node in $\mathcal{T}$, the remaining graph of Fig. E.18 (b) is obviously connected, thus satisfying Condition ② in Theorem 3.3. Therefore, for $\mathcal{T}$ of Category 2.2, all real links incident to μ_3 are unidentifiable and the remaining links are identifiable.

4) Category 2.3.

Category 2.3 is illustrated in Fig. D.8. Due to the 2-vertex-connectivity of the parent biconnected component of $\mathcal{T}$, there exist pairwise internally vertex disjoint

⁸The link can be a path from neighboring biconnected component for virtual link replacement in $\mathcal{T}$.

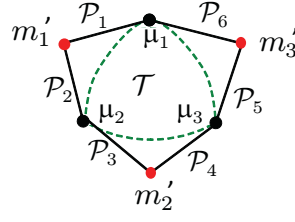


Figure E.19: Link identifications of Category 2.4.

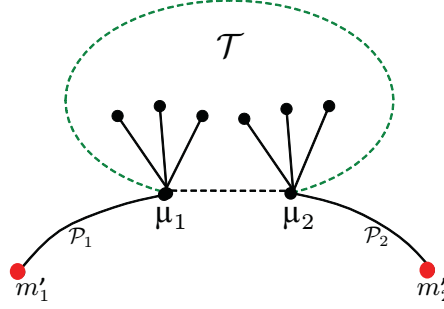


Figure E.20: Link identification of Category 3.

vantage-to-agent paths $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ and $\mathcal{P}_4$. Consider the case that $\mathcal{T}$ is 3-vertex-connected and all virtual links in $\mathcal{T}$ (except $\mu_1\mu_2$ and $\mu_1\mu_3$ if they exist) are replaced by the corresponding real paths in neighboring components by Claim 1. Then abstracting $\mathcal{P}_1, \dots, \mathcal{P}_4$ as single links, the graph in Fig. D.8 satisfies the condition in Theorem 3.3 even if $\mu_1\mu_2$ or $\mu_1\mu_3$ does not exist (i.e., virtual links). Therefore, all interior links in Fig. D.8 (i.e., all links in $\mathcal{T}$) are identifiable. In the case that $\mathcal{T}$ is a triangle, the identifiability of $\mathcal{T}$ is determined by Lemma D.6.

5) Category 2.4.

Fig. E.19 illustrates triconnected component of Category 2.4. There exist pairwise internally vertex disjoint vantage-to-agent paths $\mathcal{P}_1, \dots, \mathcal{P}_6$. Replacing all virtual links in $\mathcal{T}$ (except $\mu_1\mu_2$, $\mu_2\mu_3$ and $\mu_1\mu_3$ if they exist) by the corresponding real paths in neighboring components by Claim 1 and abstracting $\mathcal{P}_1, \dots, \mathcal{P}_6$ as single links, the resulting graph in Fig. E.19 satisfies the condition in Theorem 3.7 even if $\mu_1\mu_2$, $\mu_2\mu_3$ or $\mu_1\mu_3$ does not exist (i.e., virtual links). Therefore, all links in $\mathcal{T}$ of Category 2.4 are identifiable.

6) Category 3.

Suppose $\mathcal{T}$ contains vantages μ_1 and μ_2 , as illustrated Fig. E.20. There exist vertex disjoint paths $\mathcal{P}_1$ and $\mathcal{P}_2$ in the parent biconnected components. Thus, it is equivalent to the case of Fig. 5.5. By Claim 1, all virtual links in $\mathcal{T}$ (except $\mu_1\mu_2$) can be replaced by the corresponding real paths in neighboring components; therefore, for $\mathcal{T}$, all real links incident to μ_1 and μ_2 (except for link $\mu_1\mu_2$ if any) are unidentifiable and the remaining links are identifiable. Moreover, the identifiability of $\mu_1\mu_2$ (if it exists) is determined by Lemma D.7.

In DAIL, when triconnected component $\mathcal{T}$ is of Category 2, satisfying the conditions in lines 5 and 6 (of DAIL), then this is actually the case of Category 2.2, which is processed by line 7 (in DAIL). For the case that 3-vertex-connected $\mathcal{T}$ is of Category 2.1, 2.3 and 2.4, the above proof shows that all links in $\mathcal{T}$ are identifiable (processed by line 9 in DAIL). When a Category 2 component is a triangle, the identifiability of this component is determined by Algorithm A.3 (in line 12 of DAIL).

Given the fact that the triconnected component decomposition is not unique, next, we prove that link identifiability determined by DAIL does not depend on the mechanism for triconnected component decomposition. According to Theorem D.8, all triconnected components are unique except for the triangle components. Therefore, for a link l in $\mathcal{G}$, under a given graph decomposition, if l falls into a 3-vertex-connected or single-link component $\mathcal{T}$, then l is always in this component $\mathcal{T}$ under all other decomposition mechanisms, since $\mathcal{T}$ is unique. For such $\mathcal{T}$, if it contains unidentifiable links, then $\mathcal{T}$ must be of Category 2.2 or 3. For these unidentifiable links in $\mathcal{T}$, if they can be identified in neighboring triconnected components, then it implies that there are some other vantage-to-agent connections that are not considered for identifying $\mathcal{T}$. Furthermore, these additional vantage-to-agent connections can convert $\mathcal{T}$ to other triconnected component categories that are completely identifiable, contradicting the fact of $\mathcal{T}$ being of Category 2.2 or 3. Therefore, for a

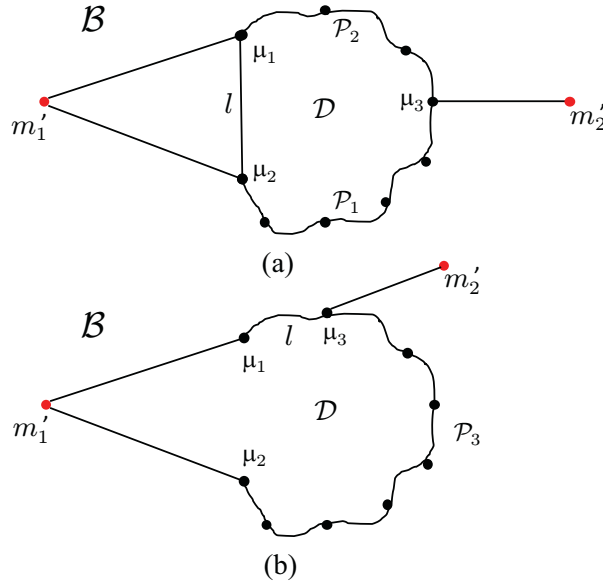


Figure E.21: Link identification of Category 2.2 in different triconnected components.

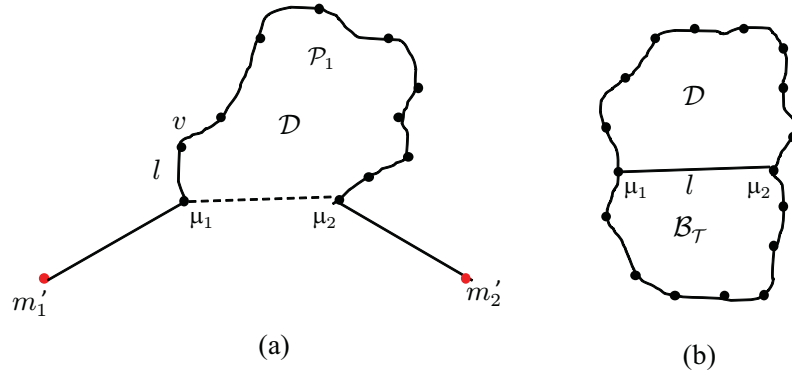


Figure E.22: Link identification of Category 3 in different triconnected components.

3-vertex-connected or single-link component $\mathcal{T}$, it is impossible that links marked as unidentifiable in $\mathcal{T}$ can be identified in neighboring triconnected components. Next, it suffices to show that the identifiability of links within triangle components remains the same for all possible graph decompositions. Consider link l within a triangle component $\mathcal{T}$ under one graph decomposition. If $\mathcal{T}$ contains 3 real links, then $\mathcal{T}$ is also unique under all possible graph decompositions (see Lemma 2 in [51]). For these triangles with 3 real links, above argument for 3-vertex-connected or single-link component also applies. Therefore, we finally consider triangle component $\mathcal{T}$ involving virtual links. Then there are 5 cases.

(6.i). $\mathcal{T}$ is of Category 2.1. In this case, $\mathcal{T}$ is identified by Algorithm A.1 (the correctness is shown in Lemma D.5). Throughout the proof of Lemma D.5, we consider the identification of l ($l \in L(\mathcal{T})$) only in its parent biconnected component, instead of triconnected component $\mathcal{T}$, e.g., v_1v_2 in Fig. D.7 (a) can be classified into multiple triangle components; however, its parent biconnected component is unique. Therefore, the generated triconnected component $\mathcal{T}$ involving l under different decomposition mechanisms does not affect the identifiability of l ;

(6.ii). $\mathcal{T}$ is of Category 2.2. This case is shown in Fig. E.21. Note all links in Fig. E.21 are real links. If $l = \mu_1\mu_2$ as shown in Fig. E.21 (a), then within the parent biconnected component, there exist real paths $\mathcal{P}_1$ and $\mathcal{P}_2$, connecting μ_3 to μ_2 and μ_1 , respectively. Abstracting $\mathcal{P}_1$ and $\mathcal{P}_2$ as single links, l is always identifiable since l is a cross-link (defined in Section 3.2.2.1). When $\mathcal{D}$ in Fig. E.21 (a) is a polygon, there are multiple triangle decompositions for $\mathcal{D}$; nevertheless, the detailed triangle decomposition does not prevent l in Fig. E.21 (a) from being a cross-link. If $l = \mu_1\mu_3$ (or $l = \mu_2\mu_3$) as shown in Fig. E.21 (b), then similar to Fig. E.21 (a), within the parent biconnected component, there exists path $\mathcal{P}_3$, connecting μ_2 to μ_3 (Fig. E.21 (b)). Abstracting $\mathcal{P}_3$ as a single link, both l and $\mathcal{P}_3$ are exterior links, thus unidentifiable. It is possible that $\mathcal{D}$ in Fig. E.21 (b) is a polygon, then there are multiple triangle decompositions for $\mathcal{D}$; nevertheless, the detailed triangle decomposition does not affect the identifiability of l in Fig. E.21 (b);

(6.iii). $\mathcal{T}$ is of Category 2.3. In this case, $\mathcal{T}$ is identified by Algorithm A.2 (the correctness is shown in Lemma D.6). Throughout the proof of Lemma D.6, we consider the identification of l ($l \in L(\mathcal{T})$) only in its parent biconnected component, instead of triconnected component $\mathcal{T}$, e.g., $\mu_2\mu_3$ in Fig. D.9 (b) can be classified into multiple triangle components; however, its parent biconnected component is unique. Therefore, the generated triconnected component $\mathcal{T}$ involving l under different decomposition mechanisms does not affect the identifiability of l ;

(6.iv). $\mathcal{T}$ is of Category 2.4. This case is shown in Fig. E.19, where each identifiable link $\mu_1\mu_2$, $\mu_1\mu_3$, or $\mu_2\mu_3$ is a cross-link. The property of being a cross-link does not depend on how triconnected component $\mathcal{T}$ is generated; therefore, the identifiability of l within $\mathcal{T}$ remains the same for all possible graph decompositions;

(6.v). $\mathcal{T}$ is of Category 3. This case is shown in Fig. E.22, where $\mathcal{D}$ is 2-vertex-connected and all links are real except that $\mu_1\mu_2$ possibly is a virtual link. In Fig. E.22 (a), if $l = v\mu_1$, then l is always unidentifiable irrespective of the graph decomposition of $\mathcal{D}$ ($\mathcal{D}$ is a polygon). If $l = \mu_1\mu_2$ as shown in Fig. E.22 (b), then the identifiability of l is determined by neighboring components according to Algorithm 5.3. Within the same parent biconnected component, let $\mathcal{B}_{\mathcal{T}}$ be the neighboring biconnected component connecting to $\mathcal{D}$ by $\{\mu_1, \mu_2\}$ (see Fig. E.22 (b)). In $\mathcal{B}_{\mathcal{T}}$, for a triconnected component involving l , if this triconnected component is of Category 1, 2.1, 2.2, 2.3, or 2.4, then the arguments in (i)–(iv) and Theorem D.8 have proved the identifiability of l is independent of the graph decomposition of $\mathcal{B}_{\mathcal{T}}$; however, if this triconnected component is of Category 3, then the identifiability of l in Fig. E.22 (b) relies on the external agent connections to neighboring biconnected components according to Algorithm 5.3, and thus the identifiability of l only depends on the biconnected component decomposition, which is unique;

Consequently, with the complete coverage of three categories, the identification efficacy of each category, and independence to graph decompositions, DAIL can determine all identifiable/unidentifiable links. $\square$

E.5 Proof of Theorem 6.1

It is easy to see from rules (i)–(iv) (stated in Section 6.1) that MMP only deploys monitors when needed, and thus no algorithm can achieve identifiability with fewer monitors. Therefore, it suffices to prove the sufficiency of the monitor placement

computed by MMP (Algorithm 6.1), i.e., all the link metrics can be uniquely identified from measurements between the selected monitors. The idea of our proof is to show that the resulting extended graph $\mathcal{G}_{ex}$ satisfies the condition in Theorem 3.7.

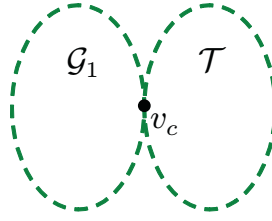
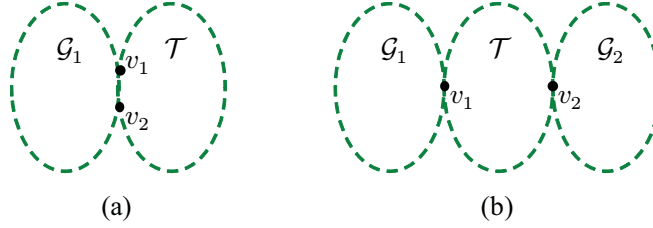
If $\mathcal{G}$ is already 3-vertex-connected, then $\mathcal{G}_{ex}$ is always 3-vertex-connected as long as there are at least three monitors, no matter how they are placed. This case is handled by lines 14–16 in Algorithm MMP. Below we will show that even if $\mathcal{G}$ is not 3-vertex-connected, Algorithm MMP still guarantees that $\mathcal{G}_{ex}$ is 3-vertex-connected. We prove this statement by showing that after removing any two nodes u_1, u_2 in $\mathcal{G}$, each remaining node in $\mathcal{G} - u_1 - u_2$ is connected to at least one monitor. There are three possible cases: 1) u_1, u_2 belong to the same triconnected component; 2) u_1, u_2 belong to different triconnected components within the same biconnected component; 3) u_1, u_2 belong to different biconnected components. We now analyze these cases separately.

1) Consider deleting two nodes u_1, u_2 in a triconnected component $\mathcal{T}$. If $|\mathcal{T}| = 2$, then $\mathcal{T}$ is a bridge. According to rules (i)–(iv) (see Section 6.1), each of the neighboring components of $\mathcal{T}$ must contain at least one monitor other than u_1 and u_2 . Thus, after deleting u_1 and u_2 , each connected component must contain at least one monitor.

Now consider the case of $|\mathcal{T}| \geq 3$. Since $\mathcal{T} - u_1 - u_2$ must be connected (due to its triconnectivity), it suffices to show that $\mathcal{T} - u_1 - u_2$ contains or is connected to at least one monitor. There are four possibilities depending on the number of separation vertices in $\mathcal{T}$, denoted by $s_{\mathcal{T}}$:

- (a) If $s_{\mathcal{T}} = 0$, then $\mathcal{G}$ must be 3-vertex-connected or a triangle. After removing two vertices, the remaining graph is either connected with at least one monitor, or a degenerate graph of a single node (which is a monitor).
- (b) If $s_{\mathcal{T}} = 1$, then there exists one cut-vertex v_c in $\mathcal{T}$ (shown⁹ in Fig. E.23). If

⁹Note that there may be more than one triconnected sub-graph connecting at v_c , but we only illustrate the minimum for clarity; the same applies to subsequent illustrations in this proof.

Figure E.23: Possible scenarios for a triconnected component $\mathcal{T}$ with $s_{\mathcal{T}} = 1$.Figure E.24: Possible scenarios for a triconnected component $\mathcal{T}$ with $s_{\mathcal{T}} = 2$.

- $v_c \notin \{u_1, u_2\}$, then $\mathcal{T} - u_1 - u_2$ is still connected to monitors in neighboring components via v_c ; if $v_c \in \{u_1, u_2\}$, then $\mathcal{T} - u_1 - u_2$ must contain a monitor because two of the non-separation vertices in $\mathcal{T}$ must be monitors by rule (iv).
- (c) If $s_{\mathcal{T}} = 2$, then there are two possible scenarios (shown in Fig. E.24), i.e., a 2-vertex-cut $\{v_1, v_2\}$, or $\mathcal{T}$ has two cut-vertices v_1 and v_2 . If $\{v_1, v_2\} = \{u_1, u_2\}$, then $\mathcal{T} - u_1 - u_2$ must contain at least one monitor because $\mathcal{T}$ contains at least one monitor that is not a separation vertex (by rule (iii)). If $v_2 \in \{u_1, u_2\}$ and $v_1 \notin \{u_1, u_2\}$ (or the other way), then $\mathcal{T} - u_1 - u_2$ is still connected to monitors in neighboring components via v_1 . Similarly, if $v_1, v_2 \notin \{u_1, u_2\}$, then $\mathcal{T} - u_1 - u_2$ is still connected to monitors in neighboring components via v_1 and v_2 .
- (d) If $s_{\mathcal{T}} \geq 3$ (three possible scenarios are shown in Fig. E.25), then Algorithm MM-P will not place any monitor in $\mathcal{T}$. Nevertheless, after removing two nodes u_1 and u_2 , at least one separation vertex must remain, and thus $\mathcal{T} - u_1 - u_2$ is still connected to monitors in at least one neighboring component via the remaining separation vertex.

The above shows that remaining nodes in $\mathcal{T}$ are still connected to monitors after removing two nodes u_1 and u_2 within the same triconnected component $\mathcal{T}$.

2) Now consider deleting two nodes u_1, u_2 in two different triconnected compo-

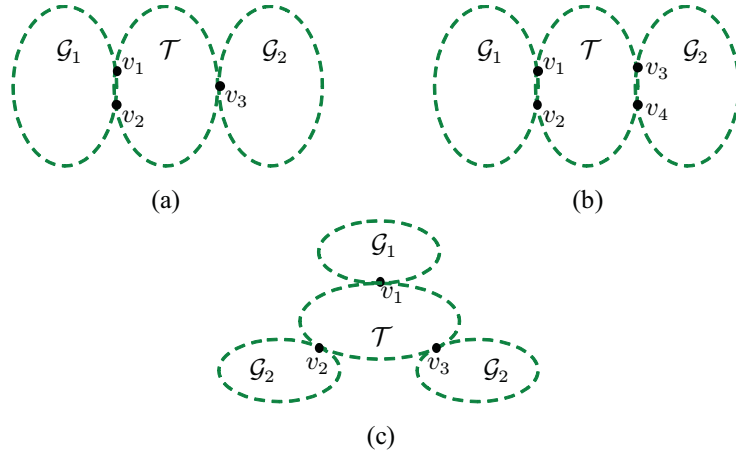


Figure E.25: Possible scenarios for a triconnected component $\mathcal{T}$ with $s_{\mathcal{T}} \geq 3$.

nents $\mathcal{T}_i, \mathcal{T}_j$ ($i \neq j$) that are in the same biconnected component $\mathcal{B}$. Note that the remaining graph $\mathcal{B}' = \mathcal{B} - u_1 - u_2$ must be connected, because the only scenario for $\mathcal{B}'$ to be disconnected is when u_1, u_2 belong to the same 2-vertex cut of $\mathcal{B}$, which implies that u_1, u_2 must belong to the same triconnected component, contradicting our assumption. Similar to Case 1), we have the following four possibilities depending on the number of cut-vertices in $\mathcal{B}$, denoted by $c_{\mathcal{B}}$:

- (a) If $c_{\mathcal{B}} = 0$, then $\mathcal{G}$ is biconnected ($\mathcal{B} = \mathcal{G}$), and $\mathcal{G} - u_1 - u_2$ must contain at least one monitor because the total number of monitors is at least 3 (see lines 14–16).
- (b) If $c_{\mathcal{B}} = 1$, then rule (iv) implies that $\mathcal{B}$ ($|\mathcal{B}| \geq 3$) contains two monitors other than the cut-vertex v_c . If $v_c \notin \{u_1, u_2\}$, then $\mathcal{B} - u_1 - u_2$ is still connected to monitors in neighboring components via v_c ; if $v_c \in \{u_1, u_2\}$, then $\mathcal{B} - u_1 - u_2$ itself contains a monitor.
- (c) If $c_{\mathcal{B}} = 2$, then $\mathcal{B}$ connects to neighboring biconnected components via two cut-vertices v_1 and v_2 . If $v_1, v_2 \in \{u_1, u_2\}$, then $\mathcal{B} - u_1 - u_2$ itself contains a monitor, as rule (iii) requires $\mathcal{B}$ to contain a monitor that is not a cut-vertex; if $v_1 \notin \{u_1, u_2\}$ (or $v_2 \notin \{u_1, u_2\}$), then $\mathcal{B} - u_1 - u_2$ is connected to monitors in at least one neighboring component via v_1 (or v_2).
- (d) If $c_{\mathcal{B}} \geq 3$, then $\mathcal{B}$ connects to neighboring biconnected components via at least three cut-vertices v_1, v_2 , and v_3 . There may not be any monitor in $\mathcal{B}$ in this case.

However, $\mathcal{B} - u_1 - u_2$ must be connected to monitors in at least one neighboring component via the remaining cut-vertex.

The above shows that nodes in $\mathcal{B} - u_1 - u_2$ are still connected to monitors after removing nodes u_1 and u_2 in two different triconnected components but within the same biconnected component $\mathcal{B}$.

Although our argument has focused on the (triconnected or biconnected) component that contains both u_1 and u_2 , it is easy to see that nodes in the neighboring components that share u_1 or u_2 are also connected to monitors after these two nodes are removed, as this is a special case of the above argument where the number of removed nodes is at most two, and the removed nodes have to be separation vertices.

Remark: The above argument assumes that virtual links have been added (see explanations after Definition 8 in Chapter 5). However, the construction of virtual links implies that if a node in $\mathcal{B} - u_1 - u_2$ is disconnected from monitors without virtual links, then this node must have only two neighbors, and line 1 in Algorithm MMP would have selected this node as a monitor. Thus, our argument holds without added virtual links.

3) Finally, consider deleting two nodes u_1 and u_2 from two different biconnected components $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. Then one of the following scenarios will occur:

- (a) If neither u_1 nor u_2 is a cut-vertex¹⁰, then $\mathcal{G} - u_1 - u_2$ is still connected. Moreover, it contains at least one monitor as there are at least 3 monitors in $\mathcal{G}$.
- (b) If u_1 is a cut-vertex but u_2 is not (or the other way), then we leverage a result from Cases 1)–2): removing two nodes from a biconnected component (whether in the same triconnected component or not), all remaining nodes are still connected to monitors. Applying this result, we see that removing one node from a biconnected component also does not disconnect any remaining node from monitors, i.e., each connected component in $\mathcal{G} - u_1$ has at least one monitor. Consider

¹⁰ u_1 and u_2 cannot be a 2-vertex cut, because otherwise u_1 and u_2 are in the same triconnected component, contradicting our assumption.

the connected component $\mathcal{G}'$ containing u_2 . Removing u_2 cannot disconnect $\mathcal{G}'$, as otherwise u_2 will have to be in the same biconnected component as u_1 or a cut-vertex itself, contradicting our assumptions; furthermore, $\mathcal{G}'$ must contain at least two monitors by rule (iv). Thus, each connected component in $\mathcal{G} - u_1 - u_2$, including $\mathcal{G}' - u_2$, has at least one monitor.

- (c) If both u_1 and u_2 are cut-vertices, then each connected component in $\mathcal{G} - u_1 - u_2$ must connect to the rest of the graph in the original $\mathcal{G}$ through one or two cut-vertices (i.e., u_1 and u_2). By rules (iii)–(iv), each of these connected components contains at least one monitor other than u_1 or u_2 .

In summary, the arguments in Cases 1)–3) imply that, for any $\mathcal{G}$ (which may not be 3-vertex-connected), Algorithm MMP guarantees that all nodes in $\mathcal{G} - u_1 - u_2$ are still connected to monitors after deleting two arbitrary nodes u_1, u_2 in $\mathcal{G}$. Since all the monitors are connected through virtual links and virtual monitors in the extended graph $\mathcal{G}_{ex}$, $\mathcal{G}_{ex} - u_1 - u_2$ must be connected. Moreover, if the virtual monitors m_1^+, m_2^+ are deleted from $\mathcal{G}_{ex}$, then the remaining graph (i.e., $\mathcal{G}$) is still connected since $\mathcal{G}$ is assumed to be connected (see Section 2.2). Finally, if m_1^+ (or m_2^+) and $u \in V(\mathcal{G})$ are deleted, then we have shown that all the nodes in $\mathcal{G} - u$ are connected to monitors, and since these monitors are still connected via m_2^+ , the entire graph $\mathcal{G}_{ex} - m_1^+ - u$ is connected. Therefore, employing Algorithm MMP to place monitors, the resulting extended graph $\mathcal{G}_{ex}$ is always 3-vertex-connected. By Theorem 3.7, this implies that the placed monitors can uniquely identify all the link metrics in $\mathcal{G}$. $\square$

