Budapest University of Technology and Economics (BME)
Faculty of Electrical Engineering and Informatics (VIK)
Department of Telecommunications and Media Informatics (TMIT)
High-Speed Networks Laboratory (HSN*Lab*)
MTA-BME Future Internet Research Group

Reliable Telecommunication Networks

D.Sc. Dissertation

In partial fulfillment of the requirements for the title of
Doctor of the Hungarian Academy of Sciences

Tapolcai János, Ph.D.

Magyar tudósok körútja 2., H-1117 Budapest, Hungary,
E-mail: tapolcai@tmit.bme.hu

Budapest
2012

Dedication

This thesis is dedicated to my wife Eszter and my son Misi. Their patience, love and endurance was essential for finishing this work.

Acknowledgements

This work was carried out at the High-Speed Networks Laboratory (HSN*Lab*) at the Department of Telecommunications and Media Informatics (TMIT), Budapest University of Technology and Economics (BME) during the years 2005–2012. I am grateful to Professor Gyula Sallai and Henk Tamás, former and current Heads of the Department, for continuously supporting my research during these years.

My deepest gratitude goes to my closest collaborators, Professor Lajos Rónyai (SzTAKI, BME) and Professor Pin-Han Ho (University of Waterloo, Canada), for the mentorship, help, advice and for those many inspiring discussions we had. Lajos's intellectual prowess is matched only by his genuinely good nature and down-to-earth humility, and I am truly fortunate to have had the opportunity to work with him. I admire Pin-Han for his dynamism, enthusiasm, creative ideas and his attitude towards science.

My warmest thanks are due to my office mates and closest co-authors, Gábor Rétvári and Péter Babarczi, for those hundreds of hours of discussions and brainstorming we had in the last five years. I am also grateful to the newly formed Lendület group with Professor József Bíró, Attila Körösi, András Gulyás, Zalán Heszberger, Felicián Németh and Balázs Sonkoly. Grateful thanks go to the Phd students I work with, Éva Hosszu, for proofreading the dissertation, and to Máté Nagy and Levente Csikor. I am grateful to all colleagues of the Lab and of the Department for the nice and inspiring atmosphere.

I wish to express my gratitude to my international collaborators Professor Muriel Medard (MIT, USA), Professor Michal Pióro (Warsaw University of Technology, Poland and Lund University, Sweden), Dirk Trossen (Cambridge University, UK), Professor Krishnaiyan Thulasiraman (University of Oklahoma, USA), Bin Wu (UESTC, China), Professor Jose L. Marzo (University of Girona, Spain) and Gergely Biczók (NTNU, Norway).

I would like to thank my PhD supervisors, Professor András Recski and Tibor Cinkler for starting my research career. I am indebted to Ferenc Nizsalovszki who was my math teacher at Fazekas Mihály Secondary School.

Very special thanks go to Ericsson, to MTA Bolyai and Magyary Zoltán Postdoctoral Scholarship for financially supporting me and my work during my postdoc research.

Last but not least, I wish to thank my wife Eszter and my son Misi for their love, patience, and endurance to my research-oriented lifestyle. I am grateful to my parents, László and Irina, for their care, and to my whole family, too. I wish to thank my lovely friends for all the fun we had together.

Contents

Chapter 1

Preface

Current Internet has reached the level of reliability where Internet (e.g. skype) and cloud (e.g. web-mail, google docs, dropbox, doodle) services have spread among users. The goal of my research is to follow this evolution and seek for solutions to reach the next level of reliability, where Internet will become a permanently operating system for the benefits of society. Fortunately, Internet was mostly built from trustworthy and intelligent equipment that could provide much more reliable services. The goal is to run the Internet for several years without any interruption of the services and slowly win the trust of most of the potential users.

The dissertation aims to provide long-term visions and some short-term solutions on the above without modifying the current IP protocol stack and keeping the correctly operating equipment. The main focus of my work is to develop efficient failure localization and restoration methods in backbone and metro networks.

1.1 Reliable Backbone Networks

We expect the services in backbone networks to further improve in the following two aspects: *flexibility and reliability*. In the near future, establishing a high speed optical connection will always be feasible within 100ms, and these connections will always operate, unless there is a catastrophe.

Backbone networks are highly vulnerable due to great physical distances. According to recent surveys, an average of 250km of optical fiber is cut once per year [1]. Fortunately, networks are designed to be self-healing against failures. They are implemented with active protection mechanisms that switch the interrupted traffic to a protection route after a failure. This switchover should be performed within 100ms to avoid any duplicated restoration attempt in the upper layers.

Since 1999 I have been dealing with survivable routing in backbone networks with *Professor Pin-Han Ho* form University of Waterloo. Our first results were published in *IEEE Trans. on Reliability* [2] and *IEEE Trans. on Networking* [3] which are among the most cited papers of this topic. As a consequence, recently I was invited to join the *IEEE INFOCOM Technical Program Committee*. Our results in the field are mainly mathematical models and routing algorithms. In *IEEE ICC'06* I have received the best paper award for modeling the service downtime distribution of the well-known survivable routing mechanisms [4]. From a theoretical point of view, optical layer restoration is the ideal protection scheme, however, it is extremely difficult to implement in a network with a fully distributed control plane, mainly because it requires fast failure localization.

In optical layer restoration at connection setup, no protection route is pre-configured, but some spare capacity on the links is reserved. After a failure, first the failed network elements are localized and protection routes are established accordingly. Restoration is considered a very advanced protection mechanism due to its simplicity at connection setup, great adaptation to sparse network topology, and efficient bandwidth utilization. It can save up to 74% protection bandwidth compared to traditional dedicated protection for single link failures and 84% for double link failures.

The first part of the dissertation (Chapters 2-4) is on fast failure localization approaches in optical networks. I started to deal with the topic in 2009 with Pin-Han Ho. I also work with *Dr. Lajos Rónyai* (MTA SZTAKI, BME) on related algebraic and graph theoretic and algorithmic problems. In failure monitoring, we have recently published more than ten top papers, three appeared in *IEEE INFOCOM* [5, 6, 7], three in *IEEE T. on Networking* [8, 9, 10] and one in *IEEE/OSA Journal of Lightwave Technology* [11]. Our main results are combinatorial algorithms with proofs on their optimality. In these proofs we often use jointly combinatorial and algebraic techniques. We are aware that some of our proofs have been already taught at graduate classes in foreign universities. Recently I was invited to summarize our results in a keynote at the *IEEE RNDM'11* conference. The key idea is to establish monitoring-trails (m-trails) in the network, which is a supervisory lightpath so the destination node will know if there is any failure along the m-trail from the interruption of the optical signal. The goal of the corresponding network design task is to set up the trails in such a way that any single failure can be unambiguously identified.

1.2 Reliable IP Networks

There are several failures that cannot be protected at the optical layer, for example the failure of an IP router. Chapter 5 deals with the failure recovery techniques at the IP layer. Several IP Fast Reroute (IPFRR) mechanisms have been proposed in the last decade; however, today Loop Free Alternate (LFA) is the only standardized and readily available IPFRR technology, mainly because LFA can be realized with straightforward modifications to current protocols, and its deployment is easy, thanks to the fact that it does not require support from other routers. This simplicity is at the expense of poor performance in terms of failure coverage. As a solution at *IEEE INFOCOM'11* [12, 13] we have proposed to add new virtual links or virtual nodes to the topology to improve the quality of LFA protection. The corresponding LFA graph extension is a challenging combinatorial problem that was further investigated in our latest study. It has won the paper award at *DRCN'11* [14]. Moreover, this concept was included in the Internet standardization and two recent IETF Internet drafts [15, 16] cite our papers. Our main results are mathematical models, algorithms and complexity bounds.

Chapter 2

Failure Localization via a Central Controller

2.1 Introduction

In transparent optical networks, failure localization is a very complicated issue that has been extensively investigated [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27].

Due to the lack of optoelectronic regenerators the impact of a failure propagates without any electronic boundary and a single failure can trigger a large number of redundant alarms [27, 28]. With failure recovery protocols at different network layers various failure management mechanisms with specific built-in failure management functionality could be adopted. Thus, a failure event at the optical layer (such as a fiber-cut) may also trigger alarms in other upper protocol layers [29], possibly causing an *alarm storm*. This not only increases the management cost of the control plane but also makes failure localization difficult. Therefore, isolating failure recovery within the network optical domain is essential to solve the problem, which will be enabled by an intelligent and cost-effective failure monitoring and localization mechanism dedicated to the network optical layer. One of the most commonly adopted approaches is to deploy optical monitors responsible for generating alarms when a failure is detected. The alarm signals then flood in the control plane of the optical network such that any routing entity can localize the failure and perform traffic restoration in a timely manner. Minimizing the number of alarm signals while achieving unambiguous failure localization (UFL) serves as the major target in the development of a failure localization scheme. In addition, reducing the bandwidth consumption for fault monitoring should also be considered.

In general, a link is a conduit of multiple fibers, and each fiber supports one or multiple wavelengths. Thus, it is intuitive to monitor a link cut event by monitoring a single wavelength along the link, and for this purpose a monitor is activated at one of the end nodes that will issue an alarm once a loss of light (LoL) is detected along the wavelength channel. This is also referred to *link-based monitoring*, which requires $|E|$ active alarms (or monitors) to monitor each link independently, where $|E|$ is the number of network links. In this case, an alarm code with a length $|E|$ is required in order to identify any single link cut event.

However, it is not considered an efficient approach to dedicatedly allocate a monitor for each link. To resolve this situation, the studies in [18, 19, 20, 21] intorduced the monitoring-cycle (m-cycle) concept, which is a pre-configured loop-back lightpath termi-

nated by an optical power monitor and launched with supervisory optical signals. When any link along the cycle is cut, the supervisory lightpath is interrupted, and the failure will be detected by the optical power monitor, and the monitor will issue an alarm to the rest of the network.

To ease the limitation on the cycle constraint of the monitoring structure, [27] introduced the concept of Monitoring-Trail (m-trail), where the model is based on an enumeration-free Integer Linear Program (ILP) approach. M-trail is proved to yield much better performance by employing monitoring resources in the shape of trails - a monitoring structure that generalizes all the previously reported counterparts. However, due to the huge computation complexity in solving the ILP, only network topologies with small sizes (such as 30 nodes) can be handled. A similar monitoring structure called "permissible probes" was considered in [30]. The study focused on theoretical proofs and asymptotic bounds, while the strength and flexibility of using tree structures for launching probes was little explored in possible practical scenarios. More detailed comparisons and descriptions of the monitoring structures (e.g. cycles, trails, trees, etc.) can be found in [31].

In this chapter, we investigate the m-trail design problem for single link and later for multi-link UFL, and aim at obtaining deeper understanding and insight into the problem. In particular, our focus is on the impact of topology diversity to the problem solutions. The chapter first provides a categorization of the current state of the art failure localization schemes. Next, it analytically derives the minimum lengths of alarm codes for several graph topologies, which is followed by an m-trail allocation algorithm developed for general topologies, which achieves a much better performance in terms of both computation time and solution quality than the ILP in [27]. It is verified by extensive simulation on thousands of randomly generated topologies.

2.1.1 Categorization of Optical Layer Failure Localization Schemes

In an optical layer monitoring scheme, a link failure is detected and localized based on the on-off status of some ligthpaths. These schemes can be categorized according to the following three aspects:

- the type of failures they can identify,
- the constraints on the lightpaths used for network status acquisition, and
- the failure localization time.

The Types of Failures

A failure could be either *hard* or *soft* [32]. A hard failure involves immediate interruption due to link and/or node function disorder typically due to fiber cuts or network node failure, while a soft failure simply degrades the performance of one or multiple wavelength channels. In this dissertation we deal with hard failures only. The failures can be further categorized according to their geographic locations. Most previous studies focused on *single-link failures*, which nonetheless account for just one third of total failures according to the network failure statistics [33]. Node failure events contribute about 20% of total failures. The rest of the failure events, including operational errors, power outage, and denial of service (DOS) attack, etc., could hit multiple links/nodes in the network simultaneously. When modeling these failures, often a list of *Shared Risk Link Groups*

(SRLGs) is defined by the network operators. An SRLG is a group of network elements subject to a risk of simultaneous failure. We call it *sparse SRLG* scenario if the number of SRLGs in the network is similar to the number of links. In this case each SRLG may contain single or multiple links and they typically correspond to a failure at a specific geographic location. For example two links over the same river share the common risk of failure because of flooding. If every single and double-link failures are part of the list of SRLGs, we call it *multiple failure (dense SRLG)* scenario. In this case there is a large number of highly overlapped and densely distributed SRLGs. Note that in both cases the SRLGs can be overlapped.

The Constraints on the Monitoring Lightpaths

Network elements can be monitored either via *in-band* or *out-of-band* monitoring [34]. The former obtains the network failure status only by monitoring the existing (or operational) lightpaths, while the latter launches supervisory lightpaths for failure status acquisition. Out-of-band monitoring is favored for its simplicity and data independence, even at the expense of more capacity consumption. Several monitoring structures, including simple/non-simple cycles, paths, non-simple trails, and bi-directional trails, etc., have been extensively studied [17, 20, 22, 23, 25, 26, 30, 35, 36, 37, 38].

First, the concept of a simple *monitoring cycle* was proposed in [19]. A simple m-cycle is a supervisory lightpath starting and terminating at the same node, which passes through each on-cycle node exactly once. Later the m-cycle concept was extended to non-simple m-cycles in [22]. In contrast to a simple m-cycle, a non-simple m-cycle is allowed to pass through a node multiple times.

Afterwards, the concept of *monitoring trails (m-trail)* was proposed [27, 5]. It differs from simple and non-simple m-cycles by removing the cycle constraint, and thus an m-trail can be taken as an acyclic supervisory lightpath with an associated monitor equipped at the destination node of the m-trail. Physical length limits on m-trails were also considered in [39].

Finally, the least constrained monitoring structure the *bidirectional m-trail (bm-trail)* was introduced in [30], where the only constraint on the set of links traversed by the lightpath is that they should be interconnected. In this case, we assume bi-directional optical links in the network, and thus they can be traversed by a directed route using the depth first search (DFS) order. Note that implementing this route as a single lightpath the nodes may require loop-back switching the optical signals coming from the transmission fiber into its reception fiber.

Failure Localization Time

The failure localization speed depends on two factors: the signaling overhead and the failure detection time for each m-trail. The latter mainly depends on the physical length of the ligthpath. As for the signaling overhead, there are three frameworks. In the first, the node where the lightpath terminates generates alarms upon any irregularity. The generated alarms are collected and used for failure localization at the node. Such alarm dissemination is at the expense of signaling overhead in the control plane, but also makes the failure localization mechanism dependent on efforts other than in the optical domain. In such a framework the goal is to achieve *Unambiguous Failure Localization (UFL)*, which means any SRLG failure can be precisely and instantly identified via monitoring a set of supervisory lightpaths.

We define a lightpath to be *local* to a node if it terminates in the node and, thus its

status can be monitored by the node. The failure localization speed can be increased if the monitoring ligthpath status information is exchanged among the fewest possible nodes. Ideally there is a single node in which every monitoring lightpath terminates. Such a node is called a *monitoring location* and is said to be capable of achieving *Local Unambiguous Failure Localization (L-UFL)*. In L-UFL after the failure is localized, the monitoring location node should broadcast this information in the network, and initiate the restoration process.

Motivated by the fact that failure localization should be carried out completely in the optical domain without taking any control plane signaling effort, a new framework was proposed recently [10, 40, 7]. It allows each node inspecting the on-off status of the monitoring lightpaths that traverse through the node, which can be done via optical signal tapping. Thus, all the nodes traversed by the monitoring lightpath can share the on-off status of the lightpath. Note that a node can only monitor the links/components of a local lightpath which are upstream to the node. Here the goal is to achieve *Network-Wide Local UFL (NL-UFL)* in the network if every node is L-UFL capable.

2.1.2 Problem Input

In a general out-of-band failure localization problem the inputs are the following.

1. The *network topology*, which is represented by an undirected graph $G = (V, E)$. The network is assumed to be 2-connected.

2. The *set of SRLGs*, which is denoted by $\mathcal{Z}$. Each SRLG $z \in \mathcal{Z}$ contains one, or multiple links.

3. The required shape of monitoring lightpath (e.g. m-cycle, m-path, m-trail, and bm-trail).

Shape Constraints

Each link e must be assigned with a binary alarm code $a_e = [a_{e,[1]}, a_{e,[2]}, \ldots, a_{e,[b]}]$, where b is the length of the alarm code. The lth binary digit, denoted by $a_{e,[l]}$, is 1 if the l^{th} monitoring ligthpath, denoted by T_l, traverses through link e, and 0 otherwise. Note that T_l has to traverse through all the links e with $a_{e,[l]} = 1$ while avoiding to take any link with $a_{e,[l]} = 0$. Conversely, let L_l denote the l^{th} *link set* which contains the set of links with $a_{e,[l]} = 1$. Depending on the *constraints on the shape* of the monitoring lightpath we have the following conditions on L_l:

***bm-trail*:** L_l must be connected;

***m-trail*:** L_l has an Euler trail, which is a connected subgraph where every node must have an even nodal degree except two: the source and destination node;

***m-path*:** L_l must be an m-trail which passes through each node only once;

***m-cycle*:** L_l must be an m-trail and the source and destination node must be the same node.

Further problem specific constraints are introduced in the next sections.

Table 2.1: Classification of hard failure localization techniques

		Hard failures		
		single link	sparse-SRLG	multiple failures
UFL	m-cycle	Sec. 2.2.1		
	m-trail	Sec. 2.2.2 and 2.2.7	[9, 38, 41]	
	bm-trail	Sec. 2.2.3 and 2.2.5	[38]	Sec. 2.3.2
L-UFL	m-cycle/path	Sec. 3.2.3	[36]	
NL-UFL	bm-trail	Sec. 3.2.4		
	in-band	[42, 43]	[32, 17]	

Table 2.2: Notation list

Notation	Description
$G = (V, E)$	undirected graph representation of the topology
$\|V\|$	the number of nodes in G
$\|E\|$	the number of links in G
b	the number of m-trails
$\mathcal{T} = \{T_1, \ldots, T_b\}$	a solution with b (b)m-trails
T_i	the i^{th} (b)m-trail, which is a set of link in G
$\|T_i\| = t_i$	the length in hops of i^{th} m-trail
$\|\mathcal{T}\| = \sum_{i=1}^{b} \|T_i\|$	the total cover length
$\underline{\underline{A}}$	the alarm code table (ACT)
a_e	the alarm code of link $e \in E$
$a_{e,[j]}$	the j^{th} bit of the alarm code of link $e \in E$
L_j	the set of links with "1" in the j-th bit position

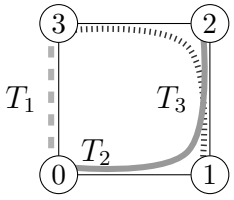
General Target Function

The cost function in out-of-band monitoring is typically composed of two ingredients:

***Monitoring cost*,** denoted by b, which is the number of monitoring lightpaths, reflects the fault management complexity. A smaller number of monitoring lightpaths results in shorter alarm codes, which further affects the number of alarms flooded in the network when a failure event occurs. In addition to larger fault management cost, a longer alarm code may cause a longer failure recovery time since a network entity has to collect all the necessary alarm signals for making a correct failure localization decision.

***Bandwidth cost*,** denoted by $\|\mathcal{T}\|$, which reflects the additional bandwidth consumption for monitoring. It is also called the *cover length* which is the sum of the lengths of each monitoring lightpath in the solution. The *length* of a monitoring lightpath is often taken as the number of links traversed by the lightpath. In this case the bandwidth cost is nothing but the total number of one bits in the alarm codes assigned to the links.

Table 2.1 indicates the sections dealing with each sub-problem.



(a) Topology and m-trails

SRLG	T_1	T_2	T_3
$(0,1)$	0	1	0
$(1,2)$	0	1	1
$(2,3)$	0	0	1
$(3,0)$	1	0	0

(b) Alarm code table (ACT)

Figure 2.1: Unambiguous failure localization (UFL) based on m-trails. The solution has $b = 3$ and $\|\mathcal{T}\| = 5$.

2.2 Unambiguous Failure Localization (UFL) for Single Failures

2.2.1 Problem Definition

The *UFL constraint* requires every link alarm code to be unique and we also have the shape constraints described in Section 2.1.2. In most of the previous works [5, 27, 22], the target was to minimize the weighted sum of the *monitoring cost* and *bandwidth cost*, formally

$$\mathcal{C} = \gamma \times (\#\ \textit{of m-trails}) + \textit{cover length} = \gamma b + \|\mathcal{T}\| \tag{2.1}$$

where $\mathcal{C}$ denotes the total cost of the solution.

In more theoretical studies on bm-trails the bandwidth cost is usually ignored, and only the number of bm-trails is considered. Some studies take out-of-band monitoring problems a network dimension problem, and take the capital expenditures instead of operating expenses [37]. In those studies the main cost is the number of transmitters.

An UFL Example

Fig. 2.1 shows an example of an UFL m-trail solution to the network in 2.1(a) for localizing any single link failure, and its *alarm code table* (ACT) is shown in Fig. 2.1(b). The ACT stores the alarm code of each link (e.g., link $(1,2)$ is assigned the alarm code 011), which further defines how the three m-trails (i.e., T_1, T_2, and T_3) should be routed. Each row of the ACT should be unique to achieve Unambiguous Failure Localization. Here, T_j has to traverse through all the links with the jth bit of the alarm code "1" while avoiding to take any link with the jth bit of its alarm code "0". By reading the status of the three m-trails, any link failure can be unambiguously localized. For example, the darkness of T_2 and T_3 depicts the failure of link $(1,2)$.

UFL for Single Failure with M-Cycles

To distinguish the failure of two links adjacent with a degree-2 node v, we need a monitoring lightpath that terminates in v, which is clearly not possible with m-cycles. Since network topologies often have degree-2 nodes, most of the recent papers deal with more relaxed shape constraints. The ILP (Integer Linear Program) for optimal design was formulated in [19, 22].

2.2.2 Lower and Upper Bounds on the Number of (B)M-Trails

The theoretical lower bound on the number of m-trails is

$$b \geq \lceil \log_2 (|E| + 1) \rceil,$$

since there are $|E|$ single failure states plus the no failure state, and the 2^b potential alarm codes must distinguish these, giving that $2^b \geq |E| + 1$.

Table 2.3 summarizes the best known lower and upper bounds on the number of (b)m-trails reported in the literature for several special graphs. For ring topologies, the number of optimal bm-trails is exactly $\lceil |E|/2 \rceil$, which was proved first in [30], and later for m-trails in [8]. Note that in ring topologies each bm-trail should only be a simple path.

The study [30] developed a construction for any graph which contains two edge disjoint spanning trees, where an upper bound of $2 \cdot \lceil \log_2 |E| \rceil - 1$ bm-trails can be achieved. The key idea of the construction is to categorize the links in the topology into two disjoint sets E_1 and E_2 of similar sizes, where $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \emptyset$, and each set contains a spanning tree. We shall generate alarm codes of length $\lceil \log_2 |E_1| \rceil + \lceil \log_2 |E_2| \rceil$ for the links in E. The links in E_1 will have unique codes in the first $\lceil \log_2 |E_1| \rceil$ bits, and similarly the links in E_2 are coded uniquely in the last $\lceil \log_2 |E_2| \rceil$ bits. At this point every link has a unique alarm code, irrespective of the values in the bits in the alarm code that has not yet been specified. These unspecified bits can be used to make the resulting test sets connected and form bm-trails. Finally, we add one additional bm-trail covering every link in E_1 and none in E_2, which can identify if the failed link belongs to E_1 or E_2. In such a way, each link has a unique alarm code with a length:

$$\lceil \log_2 |E_1| \rceil + \lceil \log_2 |E_2| \rceil + 1 \approx \lceil \log_2 \frac{|E|}{2} \rceil + \lceil \log_2 \frac{|E|}{2} \rceil + 1 = 2 \cdot \lceil \log_2 |E| \rceil - 1 \quad (2.2)$$

We refine this idea further in Section 2.2.3.

Nash-Williams and Tutte [44] showed that every $2k$-connected graph[1] has k link-disjoint spanning trees. Note that the disjoint spanning trees can be found in $O(|V||E| \log \frac{|E|}{|V|})$ time [45]. As a result, every 4-connected graph has 2 link-disjoint spanning trees, thus the proof is valid for complete graphs with more than 5 nodes[2]. For 2-dimensional square grid lattices, on the other hand, a similar technique was developed in [30] which results in $2 + 6 \cdot \lceil \log_2(n+1) \rceil$ as an upper bound on the number of bm-trails, where the graph has $(n+1) \times (n+1)$ nodes. In fact due to $|E| = 2n^2 + 2n < 2(n+1)^2$ in a square grid lattice, it leads to

$$2 + 6 \cdot \lceil \log_2(n+1) \rceil \lessapprox 2 + 6 \lceil \log_2 \sqrt{\frac{|E|}{2}} \rceil \approx 3 \lceil \log_2 |E| \rceil,$$

which is about 3 times of the theoretical lower bound: $\lceil \log_2 (|E| + 1) \rceil$.

In Section 2.2.7 an observation made from extensive simulations on thousands of general topologies is that, the m-trail solution on a topology without degree-2 nodes can achieve the theoretical lower bound of $1 + \lceil \log_2 (|E| + 1) \rceil$ provided sufficient running time for the construction. This was disproved by an example in Section 2.2.2. In this section we show a suite of polynomial-time deterministic constructions toward optimal (or essentially optimal) solutions for the (b)m-trail UFL problem.

Ring Networks

The lower bound on the number of m-trails in ring network is proved as follows [30].

A ring is a network on vertices $v_1, \ldots v_n$ whose edges (links) are $(v_1, v_2), (v_2, v_3), \ldots,$ (v_{n-1}, v_n), (v_n, v_1). Here n is the length of the ring.

[1] There does not exist a set of 2k-1 edges whose removal disconnects the graph.

[2] Based on a similar approach, an upper bound $(6 + \lceil \log_2 (|E| + 1) \rceil)$ for the m-trail formation problem is proved in Section 2.2.2.

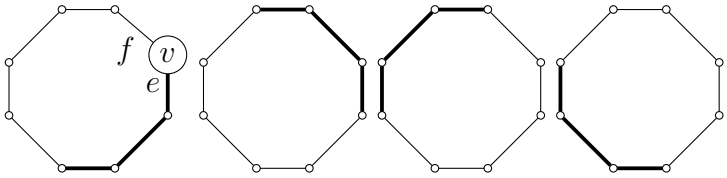
Topology	shape	lower bound	upper bound	
			in [30]	in the thesis
ring	bm-trail	$\lceil \lvert E\rvert/2\rceil$	$\lceil \lvert E\rvert/2\rceil$	Thm. 1
graph without degree-2 nodes	bm-trail	$\frac{\lvert E\rvert}{12}$, Thm. 2		
2D grid	bm-trail	$\lceil \log_2(\lvert E\rvert+1)\rceil$	$\approx 3\lceil \log_2 \lvert E\rvert\rceil$	tight + 3, Thm. 6
well connected	bm-trail	$\lceil \log_2(\lvert E\rvert+1)\rceil$	$\approx 2\lceil \log_2 \lvert E\rvert\rceil - 1$	tight, Thm. 4
fully connected	m-trail	$\lceil \log_2(\lvert E\rvert+1)\rceil$	$\approx 2\lceil \log_2 \lvert E\rvert\rceil - 1$	tight + 4, Thm. 3
Chocolate bar	m-trail	$\lceil \log_2(\lvert E\rvert+1)\rceil$		tight + 0.42, Thm. 5
$C_{1,2}$ circulant	m-trail	$\lceil \log_2(\lvert E\rvert+1)\rceil$		tight, Thm. 7

Table 2.3: The best known lower and upper bounds on the number of (b)m-trails of different graphs.

Theorem 1. *A ring topology of more than 4 nodes needs* $\lceil |E|/2\rceil$ *m-trails for single link UFL.*

Proof. We divide the proof into two claims: (1) a ring topology needs at least $\lceil |E|/2\rceil$ m-trails for single link UFL, and (2) a ring topology needs no more than $\lceil |E|/2\rceil$ m-trails for single link UFL.

[Proof of claim (1)] Let e and f be two links with a common adjacent node v, as shown on Fig. 2.2. In order to unambiguously identify failure between these two links, there must be an m-trail that passes through link e but not link f (or vice versa). Since v has degree two, this can only happen if an m-trail terminates at node v. It is clear that in a ring topology, a number of $|E|$ adjacent link-pairs can be found, and each m-trail has two terminating nodes. Therefore, it requires at least $\lceil |E|/2\rceil$ m-trails to achieve that all the n nodes are endpoints of an m-trail.



(a) M-trail T_1 (b) M-trail T_2 (c) M-trail T_3 (d) M-trail T_4

Figure 2.2: Optimal M-trail assignment of an 8-node ring.

[Proof of claim (2)] In a ring topology, every single link failure can be unambiguously identified in such a way that each m-trail is 3-hop in length and overlaps with its two neighbor m-trails by one hop, as shown in Fig. 2.2. If the ring has an odd number of nodes, the last m-trail must be a 2-hop m-trail. Thus, the network needs up to $\lceil |E|/2\rceil$ m-trails for achieving single link UFL. □

Lower Bound on the Number of M-Trails in General Graphs without any Degree-2 Nodes

Theorem 1 can be extended to the scenario of general Euler graphs in the derivation of an upper bound on the number of m-trails. Next let us give a lower bound on the number of m-trails in some "bad" two-edge-connected graphs. Clearly we have the lower bound

of $\lceil \log_2(|E|+1) \rceil$ due to the binary coding mechanism, which accounts for the fact that it takes $\lceil \log_2(|E|+1) \rceil$ bits to unambiguously identify $|E|$ different states (if "000...0" is not considered). In the following paragraphs we will demonstrate another lower bound on the number of m-trails of two-edge-connected topologies that works in parallel with the lower bound by $\log_2(|E|+1)$.

Assume that we have a set of node-disjoint graphs $G_1, G_2, \ldots, G_n$. Let the node set of G be the union of the node sets of G_i for $i = 1, \ldots n$. The edges of G are the edges of G_i and the connecting links $e_1, \ldots e_n$, where e_i connects a node of G_i to a node of G_{i+1} for $i = 1, \ldots n-1$, and e_n connects a node of G_n to a node of G_1. Clearly G is a 2-edge-connected graph if each G_i is 2-edge-connected. The set of edges $\bar{E} = \{e_1, \ldots e_n\}$ is called the *separating set*. The edges from $\bar{E}$ are called *separating links*. In the example of Fig. 2.3b we may assume that $n = 4$, and the grey links are the separating links. We shall consider m-trails in G. We call a component G_i a *boundary* of a trail T, if T includes exactly one of the separating edges incident to G_i.

Theorem 2. *At least* $\lceil \frac{|\bar{E}|}{2} \rceil = \lceil \frac{n}{2} \rceil$ *m-trails are needed to establish single link UFL in the graph* G *above.*

Proof. First we show that any m-trail T has at most two boundary components. Indeed, contract every component G_i into a single node. This transforms G into a ring, while the image of T will still be a connected subgraph. Connected subgraphs in a ring have at most two points of degree one. This implies that T has at most 2 boundary components.

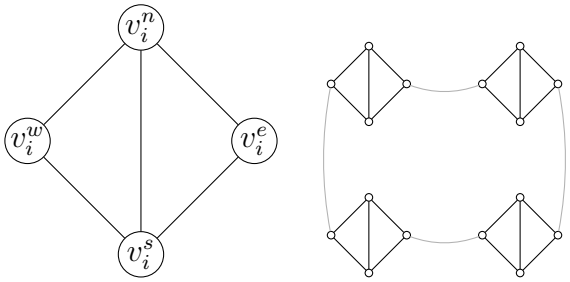
Second, we establish that every component G_i must serve as a boundary for some m-trail T. Indeed, let e and f be the separating edges incident to G_i. As the collection of our m-trails provides UFL for single link failures, there must be a trail T which contains T and does not contain e, or conversely, one which contains e but not f. In both cases G_i must be a boundary for T.

With the above two claims, we know that each m-trail has at most two boundary components, and each component must be the boundary of at least a single m-trail. Therefore, the number of m-trails in the topology is at least $\frac{|\bar{E}|}{2}$. □

With Theorem 2, the logarithmic relation between the number of m-trails and network size could be broken due to the presence of $\bar{E}$. Therefore, we can easily see that *an m-trail solution for a two-edge-connected topology with* $c+\log_2(|E|)$ *m-trails may not exist even if the topology does not contain any degree-2 node,* where c is a small positive constant. Let us define a network topology as *logarithmically proper* if an m-trail solution for the single link failure localization problem can be found with $c+\log_2(|E|)$ m-trails. Obviously, a fully meshed topology and grid topology are *logarithmically proper*, which can be covered with $c+\log_2(|E|)$ m-trails (according to the construction in Section 2.2.2 and in Section 2.2.4, respectively), while a ring topology is not (according to Theorem 1). The topology in Fig. 2.3b has $|\bar{E}| = 4$ components although without any degree-2 node, and the structure of the component as illustrated in Fig. 2.3a.The number of m-trails for the graph of Fig. 2.3b has following lower bound:

$$b \geq \frac{|\bar{E}|}{2} = \frac{|E|}{12} \tag{2.3}$$

Eq. 2.3 holds because each component (as shown in Fig. 2.3b) along with a separating link totally has 6 links, which yields $|E| = 6 \cdot |\bar{E}|$.



(a) The i-th component (b) A counter example with $|\bar{E}| = 4$

Figure 2.3: The structure of each component

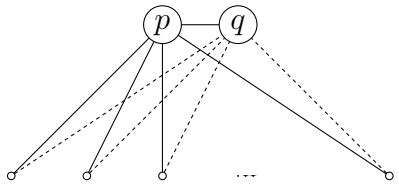


Figure 2.4: Subgraph G_p is drawn with solid lines and G_q with broken lines, while G' contains all the rest of the links of the complete graph.

Near Optimal Construction for Fully Meshed Networks

The subsection introduces a deterministic polynomial time construction of an m-trail solution for fully meshed topologies (i.e., complete graphs) that employs $4+\lceil \log_2 (|E|+1) \rceil$ m-trails for UFL. Theorem 3 validates the proposed construction. Among the 6 steps in the construction, Step (1) is for initialization, Step (2) - Step (5) are to ensure the code uniqueness of each link, and Step (6) is for m-trail formation.

Input: Complete graph $G = (E, V)$
Result: Solution with $4 + \lceil \log_2 (|E|+1) \rceil$ m-trails with $|V| \geq 7$

Step (1) Let $\mathfrak{B} = \lceil \log_2(|E|+1) \rceil$ be the theoretical lower bound on the number of m-trails. $G = (E, V)$ is first decomposed into three link-disjoint subgraphs denoted as $G' = (E', V)$, $G_p = (E_p, V)$, and $G_q = (E_q, V)$, such that p and q are two different nodes of V while E_p consists of every link adjacent to node p; and similarly E_q consists of every link adjacent to node q except the link (p, q). All the other links and nodes $v \in V \backslash \{p, q\}$ in G' form a complete graph with $|V|-2$ nodes. Thus, we have $|E_p| = |V|-1$, $|E_q| = |V|-2$, and $|E'| = |E| - 2|V| + 3$. As shown on Fig. 2.4, G_p and G_q both have the shape of a star with central nodes p and q, where $p \neq q$.

Step (2) We first allocate two m-trails, denoted as $T_{\mathfrak{B}+3}$ and $T_{\mathfrak{B}+4}$, to distinguish whether a link of G belongs to G', G_p, or G_q. As shown in Fig. 2.5, one example to achieve the above is to route the m-trail $T_{\mathfrak{B}+3}$ through all the links in $G_p \cup G_q$ while $T_{\mathfrak{B}+4}$ over all the links of G_q (and some links of G'). $T_{\mathfrak{B}+3}$ is a valid m-trail (which admits an Euler trail from p to q) because the nodal degree of each node along $T_{\mathfrak{B}+3}$ is always even except possibly at p and q. Since G_q is a star topology, the routing of $T_{\mathfrak{B}+4}$ needs some links from G' until the Euler property is met. An example of such a link set is the edge set in G' of a perfect matching.

Note that $T_{\mathfrak{B}+4}$ is used to distinguish the links in G_p from those in G_q, and $T_{\mathfrak{B}+3}$ is to distinguish links in G' from those in G_p or in G_q. Therefore with $T_{\mathfrak{B}+3}$ and $T_{\mathfrak{B}+4}$, the

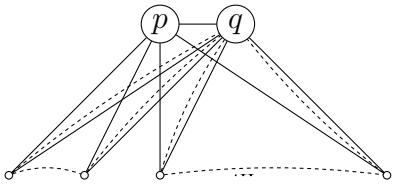


Figure 2.5: An example $T_{\mathfrak{B}+3}$ and $T_{\mathfrak{B}+4}$, where $T_{\mathfrak{B}+3}$ is drawn with solid lines and $T_{\mathfrak{B}+4}$ with broken lines.

overall UFL can be achieved provided that UFL can be achieved separately in each of the three subgraphs G', G_p, G_q. This will be done in the following steps.

Step (3) Unique non-zero binary codes of length $\lceil \frac{\mathfrak{B}+1}{2} \rceil$ bits are generated for the links in E_p. This can be done because

$$2^{\lceil \frac{\mathfrak{B}+1}{2} \rceil} \geq 2^{\frac{\mathfrak{B}+1}{2}} = \sqrt{2 \cdot 2^{\mathfrak{B}}} \geq \sqrt{2|E|} = \sqrt{2\frac{|V|(|V|-1)}{2}} > \sqrt{(|V|-1)^2} = |V| - 1. \quad (2.4)$$

The codes generated here are called *core codes* for E_p, and each of the codes serves as a $\lceil \frac{b+1}{2} \rceil$ bit-long prefix for the alarm code assigned to a link of E_p. The structure of the codes can be expressed as:

m-trails:	$T_1 \ldots T_{\lceil \frac{\mathfrak{B}+1}{2} \rceil}$	$T_{\lceil \frac{\mathfrak{B}+3}{2} \rceil} \ldots T_{\mathfrak{B}+2}$	$T_{\mathfrak{B}+3}$	$T_{\mathfrak{B}+4}$
for links in E_p	core code	$x \ldots xx$	1	0

where x denotes the yet undefined bits.

Step (4) Next unique non-zero binary codes with $\lceil \frac{\mathfrak{B}+1}{2} \rceil$ bits are generated for the links in E_q. The codes generated are called *core codes* for E_q, and each of the codes serves as a $\lceil \frac{\mathfrak{B}+1}{2} \rceil$ bit-long postfix for the alarm code assigned to each link in E_q. The structure of the codes can be expressed as:

m-trails:	$T_1 \ldots T_{\lceil \frac{\mathfrak{B}+1}{2} \rceil}$	$T_{\lceil \frac{\mathfrak{B}+3}{2} \rceil} \ldots T_{\mathfrak{B}+2}$	$T_{\mathfrak{B}+3}$	$T_{\mathfrak{B}+4}$
for links in E_q	$x \ldots xx$	core code	1	1

Step (5) Unique non-zero codes with $\mathfrak{B}+1$ bits are generated as the *core codes* for the links in E'. Note that this can easily be done since $|E| < 2^{\mathfrak{B}}$. The generated unique codes are assigned to the links in a manipulative manner described as follows. Recall that E' is a complete graph on $|V| - 2 \geq 5$ nodes. We identify two link-disjoint Hamiltonian cycles on the links of E' (e.g. by way of Walecki's construction [46, 47]), denoted by H_1 and H_2, which cover every node except $\{p, q\}$. For each link in H_1, "1" is assigned to each bit at the bit positions $1, \ldots, \lceil \frac{\mathfrak{B}+1}{2} \rceil$. Note that according to Eq. (2.4), at least $|V| - 1$ such codes exist. Similarly, for each link of H_2, "1" is assigned to each bit at the bit positions $\lceil \frac{\mathfrak{B}+3}{2} \rceil, \ldots, \mathfrak{B}+1$. The format of the codes for the links of E' is given as follows.

m-trails:	$T_1 \ldots T_{\lceil \frac{\mathfrak{B}+1}{2} \rceil}$	$T_{\lceil \frac{\mathfrak{B}+3}{2} \rceil} \ldots T_{\mathfrak{B}+1}$	$T_{\mathfrak{B}+2}$	$T_{\mathfrak{B}+3}$	$T_{\mathfrak{B}+4}$
for links in H_1	$11 \ldots 1$	code fragment		0	x
H_2	code frag.	$11 \ldots 1$	1	0	x
$E' \setminus H_1 \cup H_2$	core code in $\mathfrak{B}+1$ bits		x	0	x

Step (6) After Step (2) - Step (5), we can identify the link set L_j, $\forall 1 \leq j \leq \mathfrak{B}+2$, which contains the links with "1" in the j-th bit position in G. Let the link set contain

the links with an undefined bit at the j-th bit position be denoted as L_j^x. Now our target in this step is to extend L_j by using some of those links in L_j^x such that a valid m-trail T_j can be formed. This equivalently determines the bits of x in each link.

To ensure that L_j forms an Eulerian trail (either open or closed), we sequentially check the vertices $v \in V \setminus \{p, q\}$ to see if the degree of each v is odd or even in the current L_j. If v has an odd nodal degree, (v, q) is added to L_j if $j \leq \lceil \frac{\mathfrak{B}+1}{2} \rceil$, and (v, p) is added to L_j if $\lceil \frac{\mathfrak{B}+1}{2} \rceil < j \leq \mathfrak{B} + 2$. Therefore, we can make sure that only p and q may have an odd degree in L_j.

Then we check L_j to see if it spans a connected graph. If not, then due to the presence of one of the cycles H_1 or H_2, (p, q) is in L_j and it must be an isolated edge. In this case we simply add a link (v, p) into L_j for $v \in V \setminus \{p, q\}$ (or (v, q), respectively). The resulting graph must have an Euler trail because the odd-degree nodes must be in the set $\{v, p, q\}$.

Theorem 3. *The proposed construction on a complete graph needs no more than*

$$b = 4 + \lceil \log_2 (|E| + 1) \rceil$$

m-trails to achieve UFL for $|V| \geq 6$.

Proof. The proof of the construction is divided into two parts: (a) the code uniqueness of each link, and (b) the successful formation of an m-trail for each bit position. As for the latter, we will show that all the links with the j-th bit position as "1" are connected to form a valid m-trail, while disjoint from any link with "0" at the j-th bit position.

For part (a), the links in each subgraph have unique codes due to the intrinsic nature of the core code generation in each subgraph, which were presented in Step (3) - Step (5). Also by Step (2), the $(\mathfrak{B}+3)$-th and $(\mathfrak{B}+4)$-th bit positions are used to distinguish the links of the three subgraphs G', G_p, G_q. Therefore, the code uniqueness of each link can be ensured. For part (b), Step (6) ensures that each link set L_j are all connected with no more than 2 nodes with an odd nodal degree. Note that

$$b = \mathfrak{B} + 2 - \lceil \frac{\mathfrak{B}+1}{2} \rceil = \mathfrak{B} + 1 - \lceil \frac{\mathfrak{B}+1}{2} \rceil + 1 = \lfloor \frac{\mathfrak{B}+1}{2} \rfloor + 1 \geq \lceil \frac{\mathfrak{B}+1}{2} \rceil$$

hence for $1 \leq j \leq \lceil \frac{\mathfrak{B}+1}{2} \rceil$ the edges of G_q, while for $\lceil \frac{\mathfrak{B}+3}{2} \rceil \leq j \leq \mathfrak{B} + 2$ the edges of G_p can be used. Also note that L_j spans a connected graph on $V \setminus \{p, q\}$, due to the presence of the Hamiltonian cycles H_1 and H_2 as described in Step (5). Therefore, each $L_j, \forall 1 \leq j \leq (\mathfrak{B}+4)$, will form a valid m-trail.

With all the above, we proved that the proposed construction has each link coded with $b = (\mathfrak{B}+4)$ bits. This gives $(\mathfrak{B}+4)$ valid m-trails for achieving UFL in the fully meshed (or complete) graph G. □

Note that the proposed construction of m-trail solution for fully meshed topologies is a special case of the problem addressed in [30] by Algorithm 1, and thus it improves the $O(\log_2 |E|)$ construction (Theorem 2 of [30]) to $O(1) + \log_2 |E|$.

2.2.3 An Optimal Bm-Trail Solution in Densely Meshed Graphs

We shall need a simple inequality.

Lemma 1. *The following inequality holds for every positive integer* $b \geq 3$*:*

$$2 \cdot \lfloor \frac{2^b - 1}{b} \rfloor \geq \frac{2^{b+1} - 1}{b+1} \geq \lceil \frac{2^b - 1}{b} \rceil. \tag{2.5}$$

Proof. For the first inequality one can readily check that it holds for $b = 3, 4, 5$. Note also that the inequality fails for $b = 2$. We have

$$2 \cdot \lfloor \frac{2^b - 1}{b} \rfloor - \frac{2^{b+1} - 1}{b+1} \geq 2 \cdot \left(\frac{2^b - 1}{b} - 1 \right) - \frac{2^{b+1} - 1}{b+1}. \tag{2.6}$$

After clearing denominators, the nonnegativity of the above quantity for $b \geq 6$ is equivalent to $2^{b+1} - (2b^2 + 3b + 2) \geq 0$. But for $b \geq 4$ we have $3b + 2 < 3b + b = 4b \leq b^2$, hence it suffices to see that $2^{b+1} - 3b^2 \geq 0$, or $f(b) := \frac{2^{b+1}}{3b^2} \geq 1$.

We have $f(6) = \frac{128}{108} > 1$. Moreover, for every real $b \geq 3$,

$$\frac{f(b+1)}{f(b)} = 2 \left(1 - \frac{1}{b+1} \right)^2 \geq 2 \left(1 - \frac{1}{4} \right)^2 = \frac{18}{16} > 1,$$

It implies that $f(b) > 1$ whenever $b \geq 6$ is an integer. The second inequality holds because $\frac{2^b - 1}{b}$ is a convex increasing function, and at $b = 4$ the difference is $\frac{2^5 - 1}{5} - \frac{2^4 - 1}{4} = 2.45 > 1$. □

Next let us prove a lemma which is an important building block for the subsequent description on the proposed construction and its proof.

Lemma 2. *The nonzero binary codewords of length b can be distributed into b buckets, where the i^{th} bucket contains codewords only with 1 for the i^{th} bit, and the size of each bucket is at least $\lfloor \frac{2^b - 1}{b} \rfloor$ and at most $\lceil \frac{2^b - 1}{b} \rceil$.*

Proof. The proof is inductive, and we will give a recursive construction for such a distribution of codewords. See Fig. 2.6 as an illustration of each recursive step.

Clearly, for $b = 1, 2$ the statement trivially holds. Let us assume that the codewords of length b are already distributed into b buckets, where the i^{th} bucket has words only with 1 for the i^{th} bit, and the size of each bucket is at least $\lfloor \frac{2^b - 1}{b} \rfloor$ and at most $\lceil \frac{2^b - 1}{b} \rceil$. We define such a distribution as an *almost uniform distribution of b bits*.

Next, we consider the nonzero codewords of length $b+1$, and prove that the codewords can follow the almost uniform distribution of $b + 1$ bits. Clearly we can distribute the $2^b - 1$ codewords with 0 in the $(b+1)^{\text{th}}$ bit such that the first b bits are distributed almost uniformly (according to the given assumption under the inductive proof); namely the first b buckets are filled up with at least $\lfloor \frac{2^b - 1}{b} \rfloor$ and at most $\lceil \frac{2^b - 1}{b} \rceil$ codewords. At the end these buckets must have at least $\lfloor \frac{2^{b+1} - 1}{b+1} \rfloor$ and at most $\lceil \frac{2^{b+1} - 1}{b+1} \rceil$ codewords.

Next, let us consider the rest of the codewords. Obviously, any of them can be placed into the $(b + 1)^{\text{th}}$ bucket, because they all have 1 bit at position $b + 1$. The codeword which has 1 at the $(b + 1)^{\text{th}}$ position and 0 in the rest positions (i.e. $100 \ldots 0$) must be placed into the $(b + 1)^{\text{th}}$ bucket. The remaining $2^b - 1$ codewords can be distributed by the first b bits almost uniformly into the b buckets. In such a way, each bucket has at least $2 \cdot \lfloor \frac{2^b - 1}{b} \rfloor$ codewords, which is at least $\lceil \frac{2^{b+1} - 1}{b+1} \rceil$ according to Lemma 1. Some of the newly added codewords must be moved to the $(b + 1)^{\text{th}}$ bucket, until every bucket has at least $\lfloor \frac{2^{b+1} - 1}{b+1} \rfloor$ and at most $\lceil \frac{2^{b+1} - 1}{b+1} \rceil$. Such an action is always possible. This is argued as follows: first, codewords are moved from each of the first b buckets to the $(b+1)^{\text{th}}$ bucket so that every bucket among the first b has $\lceil \frac{2^{b+1} - 1}{b+1} \rceil$ elements. In case the $(b+1)^{\text{th}}$ bucket has less than $\lfloor \frac{2^{b+1} - 1}{b+1} \rfloor$ codewords, one more codeword from each bucket is further moved

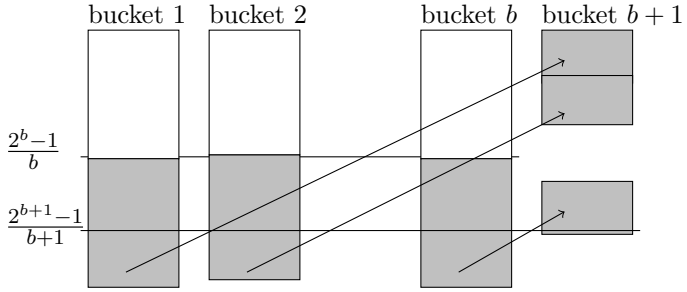


Figure 2.6: Example of the construction in the proof of Lemma 2.

to the $(b+1)^{\text{th}}$ bucket until it has $\lfloor \frac{2^{b+1}-1}{b+1} \rfloor$ codewords. Such a process will not get stuck at a position in which one bucket has less than $\lfloor \frac{2^{b+1}-1}{b+1} \rfloor$ codewords while all the others have this number, because the total number of nonzero codewords is $2^{b+1}-1$. In such a way, every bucket has at least $\lfloor \frac{2^{b+1}-1}{b+1} \rfloor$ and at most $\lceil \frac{2^{b+1}-1}{b+1} \rceil$ codewords. Thus, we proved Lemma 2. □

Theorem 4. *Let $G=(V,E)$ be a $2 \cdot \lceil \log_2(|E|+1) \rceil$ connected graph. $G=(E,V)$ can be optimally covered with $\lceil \log_2(|E|+1) \rceil$ bm-trails to achieve single-link UFL.*

Proof. Let $b = \lceil \log_2(|E|+1) \rceil$. Clearly at least b bm-trails are required for UFL in a graph with E links. In the following we will show that b is also the upper bound. Our goal for the proof of the theorem is to come up with a construction that achieves the theoretical lower bound, and then we will prove the correctness of the construction.

The Proposed Construction

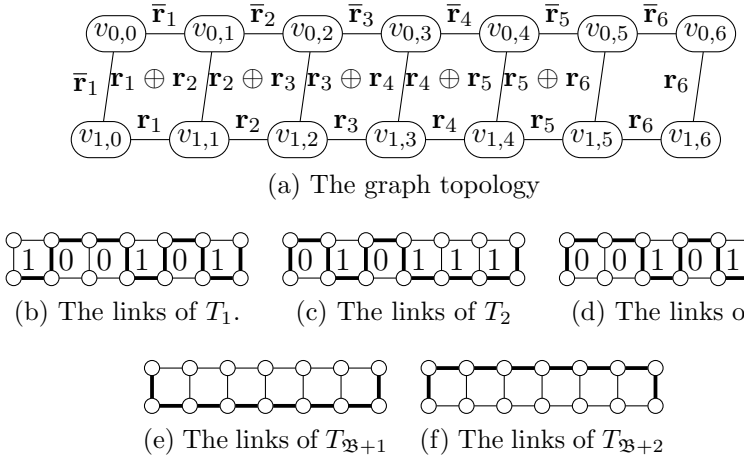
Recall that the goal of the bm-trail formation process is to assign a binary alarm code to each link so that T_i is a connected subgraph, where $i = 1, \ldots, b$. This can be ensured if each T_i has a spanning tree as a subgraph. Since every $2k$-edge-connected graph has k edge disjoint spanning trees[3] [48, 49], the construction can achieve the desired lower bound if the graph is $2 \cdot b$ connected, which is sufficient to yield $b = \lceil \log_2(|E|+1) \rceil$ edge disjoint spanning trees. Let S_i denote the i^{th} spanning tree, where $i = 1, \ldots, b$, and the spanning trees are all disjoint (i.e. $S_i \cap S_j \equiv \emptyset$ if $i \neq j$).

According to Lemma 2, the 2^b-1 nonzero codewords of b bits in length can be grouped into b buckets of size at least $\lfloor \frac{2^b-1}{b} \rfloor$, where the i^{th} bucket has alarm codes where the i^{th} bit is 1. Our construction simply assigns the codes of the i-th bucket to the i-th spanning tree S_i, while the remaining edges which are not in the $1^{\text{st}}, \ldots, b^{\text{th}}$ spanning trees, namely $E \setminus \{S_1 \cup S_2 \cup \cdots \cup S_b\}$, will be assigned the left and unused codes arbitrarily. This finishes the construction.

Correctness of the Constructed BM-Trail Solution

Since T_i contains S_i, each bm-trail must be connected and span the whole graph G. Besides, each link has a unique alarm code because nonzero unique codewords were assigned to the links of the graph. To conclude the proof we need to show that each bucket has at least $|V|-1$ codewords. By observing the equation $b \cdot (|V|-1) \leq |E| \leq 2^b - 1$, we see that each bucket has at least $|V|-1 \leq \lfloor \frac{2^b-1}{b} \rfloor$ elements. Thus, we proved Theorem 4. □

[3]Note, that such disjoint spanning trees can be found in $O(|V||E| \log \frac{|E|}{|V|})$ time [45].



(a) The graph topology

(b) The links of T_1. (c) The links of T_2 (d) The links of T_3

(e) The links of $T_{\mathfrak{B}+1}$ (f) The links of $T_{\mathfrak{B}+2}$

Figure 2.7: An example of a chocolate bar graph and the corresponding optimal solution for bm-trails. The bit of each bit position is drawn in each 1×1 rectangular. The $\mathbf{r}^1, \mathbf{r}^2, \ldots \mathbf{r}^n$ codes assigned to the links are listed in the Table 2.4

The theorem is applicable to complete graphs with at least 18 nodes because they have $\frac{18 \cdot 17}{2} = 153$ links that can be uniquely coded in 8 bits. In this case the graph is at least 16-connected.

2.2.4 An Optimal BM-Trail Solution for Chocolate Bar Graphs

Next we considers general 2D grids denoted by $S_{m,n}$, where m and n corresponds to the number of links in the vertical and horizontal direction, respectively. Harvey, *et al.* [30] provided an $3\lceil \log_2 |E| \rceil$ upper bound on the number of bm-trails according to Eq. (2.2) in the case of $m = n$.

In the next section, we generalize the study of [30] and investigate the scenario of 2D grid graphs with arbitrary m and n. We give a novel polynomial-time deterministic construction that requires no more than $3 + \lceil \log_2 (|E| + 1) \rceil$ bm-trails. We first solve the bm-trail allocation problem for a special case of $S_{m,n}$ with either $n = 1$ or $m = 1$ (called as *chocolate bar graphs*); and then a solution for general 2D grid topologies is developed based on the chocolate bar solution.

A general chocolate bar graph is denoted as $C_n(E, V)$, which has $|V| = 2n + 2$ vertices denoted as $x_{1,0}, \ldots, x_{1,n}$ (the lower points), and $x_{0,1}, \ldots, x_{0,n}$ (the upper points). Fig. 2.7(a) shows an example of a chocolate bar with $n = 6$. For link set E, we have *lower horizontal links* $(x_{1,i}, x_{1,i+1}) \in E$, *upper horizontal links* $(x_{0,i}, x_{0,i+1}) \in E$ for $i = 0, \ldots, n-1$, and the *middle vertical links* $(x_{0,i}, x_{1,i}) \in E$ whenever $i = 0, \ldots, n$.

Theorem 5. *For a chocolate bar graph $C_n(E, V)$ $b = \lceil \log_2(n+1) \rceil + 2$ bm-trails achieve single-link UFL for $b > 2$, which is at most $\lceil 0.42 + \log_2 (|E| + 2) \rceil$ bm-trails.*

Proof. The proof is developed by way of a polynomial-time deterministic construction composed of two steps. We will first introduce the construction, and then explain in detail how the construction can achieve the desired bound on the number of bm-trails.

Alarm Code Assignment for the Chocolate Bar Graph

Let us assign binary alarm codes to the links of C_n in the following way (see also Fig. 2.7). We first generate n bitvectors $\mathbf{r}^1, \mathbf{r}^2, \ldots \mathbf{r}^n$ of length $\mathfrak{B}$, where $\mathbf{r}^{i+1}$ is assigned to a lower horizontal link $(x_{1,i}, x_{1,i+1}) \in E$, where $i = 0, \ldots, n-1$. The generation of

these codes is provided in Lemma 3. On the other hand, to the higher horizontal link $(x_{0,i}, x_{0,i+1}) \in E$ we assign the bitwise complement of $\mathbf{r}^{i+1}$, denoted by $\overline{\mathbf{r}}^{i+1} = \mathbf{r}^{i+1} \oplus \mathbf{1}$ where $\oplus$ stands for the bitwise modulo 2 addition (XOR) and $\mathbf{1}$ is the all 1 vector of length $\mathfrak{B}$. Also to the middle vertical link $(x_{1,i}, x_{0,i})$ we assign the bitvector $\mathbf{r}^i \oplus \mathbf{r}^{i+1}$ for $i = 1, \ldots, n-1$. Finally to the link $(x_{1,0}, x_{0,0})$ bitvector $\mathbf{r}^1 \oplus \mathbf{1}$ is assigned, and to the link $(x_{1,n}, x_{0,n})$ we attach $\mathbf{r}^n$.

In choosing the list of bitvectors $\mathbf{r}^i$, for $i = 1, \ldots, n-1$, we make the following three assumptions:

(A1) The vectors $\mathbf{r}^i$ are pairwise different for $i = 1, \ldots, n$.

(A2) The vectors $\mathbf{r}^i \oplus \mathbf{r}^{i+1}$ are all nonzero and pairwise different for $i = 1, \ldots, n-1$.

(A3) The first bits of the vectors $\mathbf{r}^1$ and $\mathbf{r}^n$ are the same.

The following statement provides an approach to construct $n \leq 2^{\mathfrak{B}} - 1$ bitvectors $\mathbf{r}^i$ which satisfy the requirements (A1), (A2), (A3).

A Brief Introduction to Galois Fields In the arithmetic of ordinary numbers there are infinitely many numbers, while the fields $\mathbb{F}_{2^b}$ have only 2^b elements. However, the operations of addition, subtraction, multiplication and division (except division by zero) may be performed in a way that satisfies the familiar rules from the arithmetic of ordinary numbers. Concerning $\mathbb{F}_{2^b}$ a widely accepted approach is to represent the elements as polynomials of degree strictly less than b over $\mathbb{F}_2$. Operations are then performed modulo R where R is an irreducible polynomial of degree b over $\mathbb{F}_2$. For example the field $\mathbb{F}_8$ can be interpreted as the polynomials modulo $1 + x + x^3$. This way we can consider $\mathbb{F}_8$ as the set of binary polynomials of degree at most 2 (indeed there are 8 such polynomials). Addition is the usual binary polynomial addition. For example

$$(1 + x + x^2) + (1 + x^2) = x.$$

Multiplication is the usual polynomial multiplication, followed by reduction if necessary (modulo $1 + x + x^3$). By reduction we mean replacing x^3 by $x + 1$ as long as it is possible. For example

$$(1 + x + x^2)x^2 = x^2 + x^3 + x^4 = x^2 + (x + 1) + x(x + 1) = x^2 + x + 1 + x^2 + x = 1.$$

In this representation x is a *primitive element*, indeed 7 is the smallest positive integer exponent m for which $x^m = 1$.

Lemma 3. *Let* $\mathfrak{B} := \lceil \log_2(n+1) \rceil$ *and* $\mathfrak{B} > 2$. *Then a series of* $n \leq 2^{\mathfrak{B}} - 1$ *nonzero codes* $\mathbf{r}^1$, $\mathbf{r}^2$, ..., $\mathbf{r}^n$ *can be generated in polynomial time to satisfy properties (A1), (A2) and (A3).*

Proof. With $\mathfrak{B} := \lceil \log_2(n+1) \rceil$, $q = 2^{\mathfrak{B}}$ is the smallest power of 2 which is greater than n. Following the widely used technique in classical error correcting codes, our code vectors will be vectors from a linear space over the two element field $\mathbb{F}_2$. We shall consider the finite (Galois) field $\mathbb{F}_q$ with q elements.

According to Theorem 2.5 in [50], $\mathbb{F}_q$ always exists and it forms a vector space of dimension $\mathfrak{B}$ over its subfield $\mathbb{F}_2$. This way we can identify $\mathbb{F}_q$ with bit vectors of length $\mathfrak{B}$, where the all zero vector corresponds to the 0 element of $\mathbb{F}_q$. In particular, nonzero

Table 2.4: The nonzero elements of $\mathbb{F}_8$ as binary polynomials modulo $1 + x + x^3$.

Exponential	Polynomial	Code
α^0	1	$\mathbf{r}^1 = 100$
α^1	x	$\mathbf{r}^2 = 010$
α^2	x^2	$\mathbf{r}^3 = 001$
α^3	$x^3 = 1 + x \mod 1 + x + x^3$	$\mathbf{r}^4 = 110$
α^4	$x + x^2$	$\mathbf{r}^5 = 011$
α^5	$x \cdot (x + x^2) = 1 + x + x^2 \mod 1 + x + x^3$	$\mathbf{r}^6 = 111$
α^6	$x \cdot (1 + x + x^2) = 1 + x^2 \mod 1 + x + x^3$	$\mathbf{r}^7 = 101$

vectors correspond to the nonzero elements of the field. Also, according to Theorem 2.8 in [50], $\mathbb{F}_q$ contains a primitive element α, which is a nonzero element such that all the powers $\alpha = \alpha^1, \alpha^2, \ldots, \alpha^{q-1}$ are pairwise different. See also Table 2.4 where the elements and the related codes are listed for $q = 8$ ($\mathfrak{B} = 3$).

Finding a primitive element in $\mathbb{F}_q$ can be done in polynomial time with exhaustive search, because any nonzero element α can be verified for being a primitive element by raising to a power and checking if the power equals to 1 with an exponent less than $q-1$.

We now set $\mathbf{r}^i$ to be the (bit vector of the) element α^{i-1}. Condition (A1) is satisfied as $n \leq 2^{\mathfrak{B}} - 1$.

Suppose now that (A2) fails. Then there must exist $0 \leq i < j < n-1$ such that $\alpha^i \oplus \alpha^{i+1} = \alpha^j \oplus \alpha^{j+1}$ holds in $\mathbb{F}_q$. But then we have $\alpha^i(1 \oplus \alpha) = \alpha^j(1 \oplus \alpha)$ which (using that $\mathfrak{B} > 1$ and hence that $1 \oplus \alpha$ is not 0) would imply that $\alpha^i = \alpha^j$, contradicting to the fact that α is a primitive element.

To establish (A3), we note that (assuming $\mathfrak{B} > 2$) α and α^n span a subspace of dimension at most 2 of $\mathbb{F}_q$ over $\mathbb{F}_2$, hence we can select the basis of $\mathbb{F}_q$ so that both element have 0 coordinates with respect to the first basis vector. □

The Proposed Construction for Chocolate Bar Graphs

In the chocolate bar construction, T_j is actually a simple path in C_n from $x_{1,0}$ to $x_{0,n}$. In the rest of the section C_n can also be denoted as $C_{x_{1,0},x_{0,n}}$. As a result, $\mathfrak{B}$ bm-trails from $x_{1,0}$ to $x_{0,n}$ are formed in C_n, each corresponding to one bit position of the vectors. An example is given with $n = 5$, where the resultant 5 m-trails by the construction are shown in Fig. 2.7(b), 2.7(c), 2.7(d).

In addition to the above mentioned bm-trails, we need to add two more bm-trails. This is exemplified in Fig. 2.7(e) and 2.7(f). Let the two bm-trails correspond to $T_{\mathfrak{B}+1}$ and $T_{\mathfrak{B}+2}$, respectively, where $T_{\mathfrak{B}+1}$ is composed of the links $(x_{1,0}, x_{0,0})$, $(x_{1,n}, x_{0,n})$ and the path consisting of all the links $(x_{1,i}, x_{1,i+1})$ $i = 0, \ldots n-1$, while $T_{\mathfrak{B}+2}$ is composed of the links $(x_{1,0}, x_{0,0})$, $(x_{1,n}, x_{0,n})$ along with the path consisting of all the links $(x_{0,i}, x_{0,i+1})$, $i = 0, \ldots n-1$. As a result, $T_{\mathfrak{B}+1}$ and $T_{\mathfrak{B}+2}$ can identify whether a failed link was a horizontal or vertical link, and whether the link was $(x_{1,0}, x_{0,0})$ or $(x_{1,n}, x_{0,n})$.

Corollary 1. *Each T_j $j = 1, \ldots, \mathfrak{B} + 2$ forms a single bm-trail, and every bm-trail is a simple path.*

The corollary clearly holds according to the construction.

Correctness of the Constructed Solution

We will show in the following paragraphs that the set of bm-trails $T_1, \ldots, T_{\mathfrak{B}+2}$ are able to localize any single link failure in chocolate bar C_n. Obviously, T_1, $T_{\mathfrak{B}+1}$ and $T_{\mathfrak{B}+2}$

can unambiguously localize any failed link among $(x_{1,0}, x_{0,0})$ and $(x_{1,n}, x_{0,n})$ because the faulty link can be one of $(x_{1,0}, x_{0,0})$ or $(x_{1,n}, x_{0,n})$ if and only if both $T_{\mathfrak{B}+1}$ and $T_{\mathfrak{B}+2}$ are faulty. If both $T_{\mathfrak{B}+1}$ and $T_{\mathfrak{B}+2}$ alarm (i.e., report failure), the status of T_1 can be used to determine which of the two links $(x_{1,0}, x_{0,0})$ or $(x_{1,n}, x_{0,n})$ is at fault according to (A3).

For the other links, the statuses of $T_{\mathfrak{B}+1}$ and $T_{\mathfrak{B}+2}$ can be used to determine whether the faulty link is in the group of lower links, the group of upper links, or the group of middle links. With (A1), the links in the first two groups are pairwise different, while with (A2) it implies that the codes in the group of middle links are pairwise different. Therefore, all the links in each of the 3 groups are distinguishable such that unambiguous failure localization is possible within each group, and hence in C_n.

The Number of BM-Trails in the Construction

Since the chocolate bar graph has $3n + 1$ links, we have

$$\mathfrak{B} = \lceil \log_2(\frac{|E| - 1}{3} + 1) \rceil < \lceil -1.58 + \log_2(|E| + 2) \rceil.$$

As a result the construction requires at most $b = \mathfrak{B} + 2 = \lceil 0.42 + \log_2(|E| + 2) \rceil$ bm-trails. □

2.2.5 An Essentially Optimal BM-Trail Solution for 2D Grid Topologies

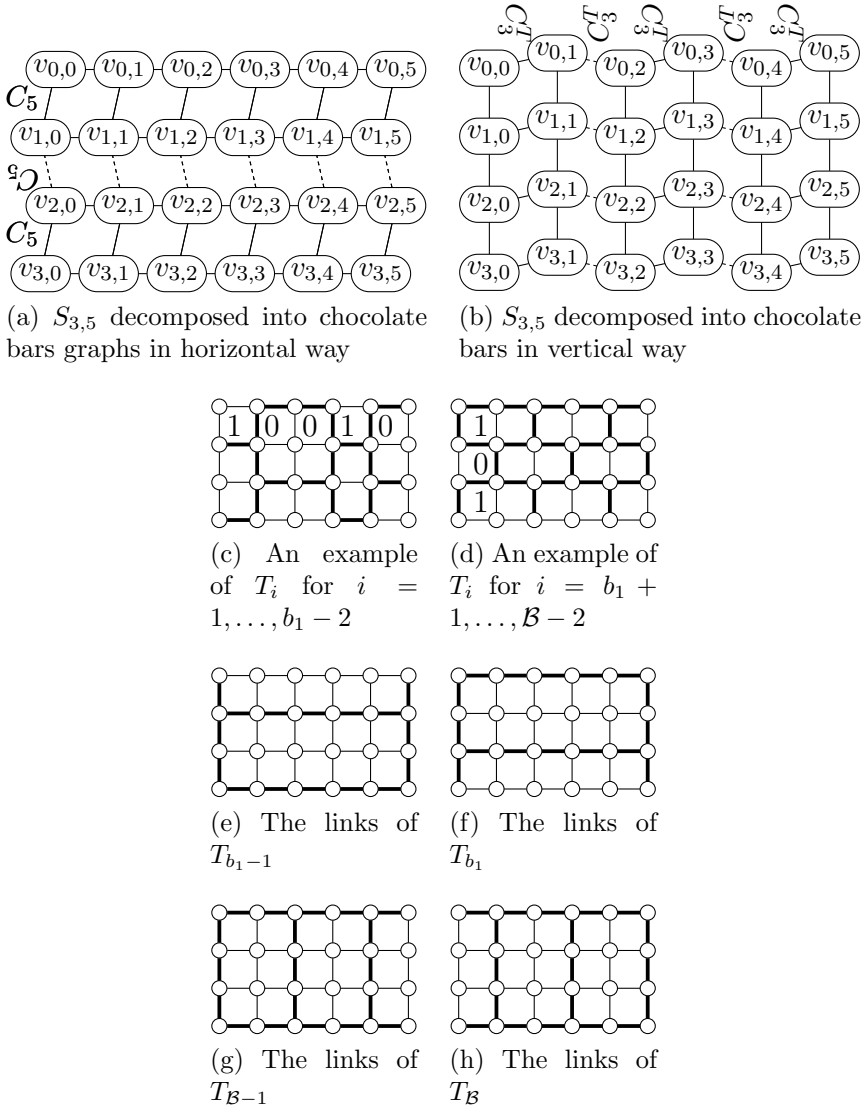
In this section, the construction for the chocolate bar graphs is generalized for 2D rectangular grids. A $(m+1)$-by-$(n+1)$ grid graph is denoted as $S_{m,n}$, whose vertices are denoted as $x_{i,j}$ for $0 \le i \le m$ and $0 \le j \le n$. The vertical links of $S_{m,n}$ are $(x_{i,j}, x_{i+1,j})$ for $0 \le i < m$ and $0 \le j \le n$. Analogously, the horizontal links of $S_{m,n}$ are $(x_{i,j}, x_{i,j+1})$ for $0 \le i \le m$ and $0 \le j < n$.

Theorem 6. *A 2D rectangular grid graph $S_{m,n}(E, V)$ can be covered with $3+\lceil \log_2(|E| + 1) \rceil$ bm-trails to achieve UFL, for $m, n \ge 1$.*

Proof. We shall have two monitoring sets of bm-trails. The first set has size $b_1 = \lceil \log_2(m + 1) \rceil + 2$, while the second has size $b_2 = \lceil \log_2(n + 1) \rceil + 2$. Informally speaking, the first set gives the horizontal position of a failed link, while the other gives the vertical coordinate. This will be sufficient to locate the failed link unambiguously. In total, we shall have no more than $\mathcal{B} = b_1 + b_2$ monitoring bm-trails.

The Proposed Construction

We are going to extend the bm-trails T_i $(i = 1, \ldots, b_1)$ from the chocolate bar graph C_n to the whole square grid $S_{m,n}$. We do it step by step as follows: first we reflect the bm-trail T_i with respect to the line connecting $x_{1,0}$ to $x_{1,n}$, such that T_i is extended to the second chocolate bar defined by the vertices $x_{1,j}$ and $x_{2,j}$, for $j = 0, \ldots, n$. The second chocolate bar is extended analogously by reflection to the third chocolate bar, defined by $x_{2,j}$ and $x_{3,j}$, and so on. This reflection process is repeated until the whole $S_{m,n}$ is covered, where the 2D rectangular grid is treated as a series of chocolate bar graphs of C_n. As shown in Fig. 2.8(a), at every second line the chocolate bar graph is upside down, and the i-th chocolate bar graph C_n consists of vertices $x_{i,0}, \ldots, x_{i,n}$ and $x_{i+1,0}, \ldots, x_{i+1,n}$, where $i = 0, \ldots, m-1$. By applying the reflection process for all the bm-trails T_i $(i = 1, \ldots, b_1)$, we will obtain b_1 bm-trails.



(a) $S_{3,5}$ decomposed into chocolate bars graphs in horizontal way

(b) $S_{3,5}$ decomposed into chocolate bars in vertical way

(c) An example of T_i for $i = 1, \ldots, b_1 - 2$

(d) An example of T_i for $i = b_1 + 1, \ldots, \mathcal{B} - 2$

(e) The links of T_{b_1-1}

(f) The links of T_{b_1}

(g) The links of $T_{\mathcal{B}-1}$

(h) The links of $T_{\mathcal{B}}$

Figure 2.8: An example of a 2D lattice graph of size 3×5.

It is clear that the result of the reflection process must be a connected subgraph without fragmentation, thereby its eligibility as a bm-trail is ensured.

With the whole situation transposed, exactly the same method is applied to specify the vertical position i of the faulty link e. For the remaining b_2 bm-trails of the rectangular grid $S_{m,n}$ we start out with the vertically placed chocolate bar C_m^T at the left end of the grid (see Fig. 2.8(b)) and extend the b_2 bm-trails of this C_m^T to the whole grid with the mirror-reflection procedure employed before, nonetheless from left to right in order to extend the bm-trails to all the vertical chocolate bars in the grid. By doing this b_2 bm-trails can be obtained.

With the b_1 and b_2 bm-trails, we complete the construction.

Correctness of the Constructed BM-Trail Solution

In case of a single failure T_{b_1-1}, T_{b_1}, $T_{\mathcal{B}-1}$, and $T_{\mathcal{B}}$ (see also Fig. 2.8(e), 2.8(f), 2.8(g), and 2.8(h)) can identify whether a horizontal or a vertical link has failed and if the link is on the left or right border of the rectangular grid (it is on the first or last row/column). Since the failed link belongs to at least one of the horizontal chocolate bar graph C_n, the

corresponding $b_1 - 2$ bm-trails can identify the column of the failed link. Similarly, the failed link belongs to at least one of the vertical chocolate bar graph C_m^T, the corresponding $b_2 - 2$ bm-trails can identify the row of the failed link. As a result, it is known if the link is horizontal or vertical, and its column and row thus can be localized.

The Number of BM-Trails in the Construction

Bm-trails T_{b_1-1}, T_{b_1}, $T_{\mathcal{B}-1}$, and $T_{\mathcal{B}}$ are only used to decide if the link is horizontal or vertical and if the link is on the boundary of the grid, which indeed can be done with only two bm-trails instead of four, namely $T_{b_1-1} \cup T_{b_1}$ and $T_{\mathcal{B}-1} \cup T_{\mathcal{B}}$. Since $S_{m,n}$ has totally $|E| = 2 \cdot m \cdot n + n + m$ links, the number of bm-trails is:

$$\begin{aligned}\mathcal{B} = \lceil \log_2(m+1) \rceil + \lceil \log_2(n+1) \rceil + 2 \leq \\ \lceil 1 + \log_2(m+1) + \log_2(n+1) \rceil + 2 = \\ \lceil \log_2 2 + \log_2(m+1) + \log_2(n+1) \rceil + 2 = \\ \lceil \log_2 2 \cdot (m+1) \cdot (n+1) \rceil + 2 = \\ \lceil \log_2 2mn + 2n + 2m + 2 \rceil + 2 = \\ 2 + \lceil \log_2(2|E| - 2mn + 2) \rceil < \\ 2 + \lceil \log_2(2|E| + 2) \rceil = 3 + \lceil \log_2(|E| + 1) \rceil \quad (2.7)\end{aligned}$$

for $m, n \geq 1$. Note that the first inequality holds because of the general inequality $\lceil A \rceil + \lceil B \rceil \leq \lceil A + B \rceil + 1$, while the second follows from $m \cdot n > 0$. □

More generally, a similar construction can be used to cover a cubic graph of any dimension for single link UFL with $O(1) + \log_2 |E|$ bm-trails. In this case the alarm code is divided into three parts, and each of them corresponds to a chocolate bar graph.

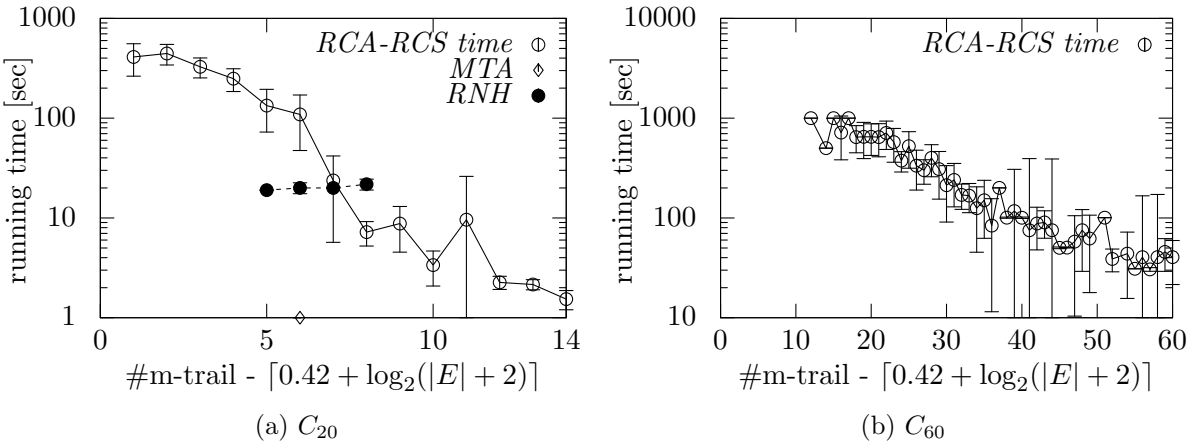
Chocolate Bar Graph as a Benchmark

Simulation is conducted on chocolate bar topologies with different number of columns, aiming to examine the performance by a number of previously reported heuristic algorithms. We will show that the heuristics perform badly in chocolate bar topologies, which nonetheless can be optimally solved with the proposed construction by a very fast algorithm.

The reported heuristic algorithms for solving the m-trail allocation problem in chocolate bars are listed in the followings.

1. *RCA-RCS* heuristic of Section 2.2.7.
2. *MTA* heuristic by [51], which is a deterministic approach that builds the m-trails in parallel until UFL is achieved.
3. *RNH* heuristic by [52] which is a randomized version of the *MTA* heuristic.
4. *Cycle Accumulation (CA)*: a generic approach by employing Dijkstra's algorithm to distinguish each pair of links [35].

We consider the three schemes: RCA-RCS, MTA, and RNH in chocolate bar graphs C_{20}, C_{40} and C_{60}. A high-performance server with 3GHz Intel Xeon CPU 5160 was used in the simulation. The result of the proposed 2D grid construction is calculated first using the theoretical optimum given in Section 2.2.4: $\lceil 0.42 + \log_2(|E| + 2) \rceil$, which yields



(a) C_{20} (b) C_{60}

Figure 2.9: Simulation results on C_{20} and C_{60} for the number of bm-trails.

the minimum number of bm-trails for UFL as 7, 7, and 8 for C_{20}, C_{40} and C_{60} with $|E| = 61$, 121, and 191, respectively. We are interested in the difference between the result by each considered heuristic and the one obtained via the proposed construction. It is important to note that RCA-RCS and RNH are both randomized approaches where longer computation time guarantees better performance (or smaller numbers of bm-trails). Therefore, we are further interested to see how much long the two schemes can converge close to the optimal solution, where the computation time describes the efficiency (and inefficiency) of the two schemes in the considered scenarios.

Fig. 2.9 demonstrates the comparison results. Clearly, both RCA-RCS and RNH show better solution quality by granting longer running time, while MTA is a deterministic algorithm that iteratively finds the longest segment as the next m-trail. Since all the three topologies are very sparse whose diameters are much longer than the average nodal degree, MTA needs 6 and 15 more bm-trails in C_{20} and C_{40} than the optimum (i.e., 7), respectively, as shown in 2.9(a) and 2.9(b). On the other hand, both RCA-RCS and RNH are seen to converge very slowly as the network has a larger diameter. As shown in Fig. 2.9(a), RCA-RCS needs to take over 400 seconds to achieve one more bm-trail away from the optimal in C_{20}, but over 800 and 1,000 seconds to achieve about 10 bm-trails from the optimal in C_{40} and C_{60}, respectively, as shown in 2.9(b) and 2.9(c).

Note that CA requires 93 bm-trails which are not shown in the figures since it is out of the range. We do not show RNH in 2.9(b) and 2.9(c), and MTA in 2.9(c), because they were not solvable in the topologies C_{20} and C_{40} by running out of 2GB memory (which is the computation specification for the simulation). This is mainly because both methods are designed for networks when the average nodal degree is not much smaller than the diameter of the network. Therefore, the two schemes had a large amount of candidate segments which drained the memory usage.

2.2.6 Optimal M-Trail Solution for Circulant graphs

A circulant $C_n(1,2)$ graph has nodes $V = \{v_0, v_1, \ldots, v_{n-1}\}$ with each node v_j adjacent to $[v_{j+1} \mod n]$ and $[v_{j+2} \mod n]$. An example of a circulant $C_9(1,2)$ is given in Fig. 2.10. Circulant graphs are considered to have similar properties to practical carrier topologies.

Theorem 7. *Circulant graph $G = C_n(1,2)$ can be covered with $b = \lceil \log_2(2n+1) \rceil$ m-trails for NL-UFL, where each m-trail is a spanning sub-graph of G, and the bandwidth cost is*

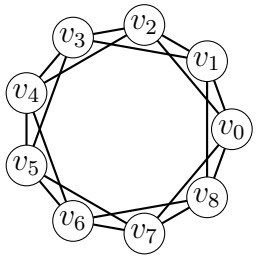


Figure 2.10: An example of a circulant graph $G = C_9(1, 2)$

$\|\mathcal{T}\| = b \cdot n + 1$.

Proof. To prove the theorem, our approach is via a novel construction that generates a set of connected subgraphs of G as m-trails, which can be proved to achieve UFL for any single-link failure.

The Proposed Construction

Let an alarm code of each link in G be b bits in length. The code $[11\ldots1]$[4] is assigned to edge (v_0, v_1), while the other edges (v_i, v_{i+1}) are each assigned with an alarm code that is the binary representation of the value $i + 1$ for $i = 0, \ldots, n - 1$. Note that, the first bit of these codes is always 0 and the rest is a nonzero bit vector. Moreover, these can be accomplished by using b bits only. Indeed, we have $2^b \geq 2n + 1$ by assumption. This implies that $2^b \geq 2n + 2$, giving that even the binary code of n bits fits in $b - 1$ bits. Edge (v_i, v_{i+2}) is assigned with a code which is bitwise complement of the alarm code of (v_i, v_{i+1}) where $i \neq 0$. Thus, the first bit of these codes for (v_i, v_{i+2}) is 1. Besides, the complementary pair of codes $[00\ldots01]$ and $[11\ldots10]$ are not assigned to any edge. Finally, edge $(0, 2)$ is associated with the bit vector $[00\ldots01]$.

The set of m-trails is deployed in such a way that T_j traverses through all the edges with their jth bit value 1 while disjoint from any edge with the jth bit value 0.

Correctness of the Constructed M-Trail Solution

It is clear that such an assignment generates a unique alarm code for each link. In the rest of the proof we show that for any $1 \leq j \leq b$, the subgraph T_j corresponding to links with their jth bit position of alarm codes 1 is connected and spans the whole vertex set. To make the proof easily presented, let us take each edge in G as directed counter-clockwise, i.e., edge (v_i, v_{i+1}) is directed from v_i to v_{i+1}, and similarly (v_i, v_{i+2}) from v_i to v_{i+2}. It is sufficient to show that every directed cycle from the edges of T_j passes through node 0. This is due to the following two facts: (1) the codes of (v_i, v_{i+1}) and (v_i, v_{i+2}) as $v \neq 0$ are bitwise complement to each other, thus T_j must connect from v_i to either v_{i+1} or v_{i+2}, this results every cycle to traverse the circulant graph counter-clockwise through either node 0 or 1; and (2) when $i = 0$, edge (v_0, v_1) is in T_j for every j, since it has a code $[11\ldots1]$. The above two facts make the outdegree in T_j of every node of G at least 1[5].

[4] $x\ldots x$ denotes a code fragment with x in every bit position

[5] To be more specific, the outdegree of v_i is exactly 1, if $i \neq 0$, and the outdegree of vertex v_0 is larger or equal to 1.

Number of M-Trails in the Construction

To evaluate the total cost, the alarm codes for the directed edges leaving vertex 0 contain exactly $b+1$ values of 1 (i.e., edge $(0,1)$ has b m-trails to traverse through, and edge $(0,2)$ has one). For the other edges (v_i, v_{i+1}) and (v_i, v_{i+2}) with $i \neq 0$, each of them has b values of 1. This implies that the total number of 1s is $bn+1$ as claimed. □

The proposed construction is optimal in terms of the number of m-trails as shown by the information theoretic lower bound $b \geq \lceil \log_2(|E|+1) \rceil$ and $|E| = 2n$.

2.2.7 The RCA-RCS Heuristic Approach for UFL

Surprisingly $\lceil \log_2(|E|+1) \rceil$ m-trails are enough in typical network topologies. The proposed algorithm takes advantage of *random code assignment* (RCA) and *random code swapping* (RCS), aiming to overcome the topology diversity. With RCA, it takes $|E|$ unique alarm codes which are randomly assigned to each link one after the other at the beginning and is kept in an alarm code table (ACT). This leads to $\lceil \log_2(|E|+1) \rceil$ link sets. The algorithm then performs m-trail formation by examining the connectivity of each link set. There could be much more m-trails than $\lceil \log_2(|E|+1) \rceil$ formed at the beginning. To improve the solution quality, RCS is performed to update the ACT for each link set round by round, where a better structure of a link set is searched according to the cost function of Eq. (2.1). Note that, RCS is performed independently (or locally) at each link set, where the codes of two links of different link sets can be swapped only if the swapping will not alter the connectivity of the other link sets. This is referred to as the *strong locality constraint* (SLC), which is an important feature of our design in making the algorithm simpler and running faster.

Fig. 2.11 shows a flowchart of the proposed algorithm. At the beginning, an alarm code table (denote by $\underline{\underline{A}}$) is formed by randomly assigning each link with a unique alarm code as shown in Step (1). In Step (2), the cost of the current $\underline{\underline{A}}$ is evaluated by Eq. (2.1) (which will be further elaborated in subsection 2.2.7). Next, a greedy cycle formed by Steps (2), (3), (4), (5), and (6) is initiated, where RCS is performed in Step (3) and (5) (which will be further detailed in subsection 2.2.7). In every cycle, a new ACT (denoted as $\underline{\underline{A}}'$) is generated and the corresponding cost $\mathcal{C}'$ is evaluated in Step (2). If the cost of $\underline{\underline{A}}'$ (denoted as $\mathcal{C}'$) is smaller than (or equal to) that of the cost of previous $\underline{\underline{A}}$ (denoted $\mathcal{C}$) as in Step (2), the algorithm starts the next greedy cycle by replacing the old ACT with the new one (i.e., $\underline{\underline{A}} \leftarrow \underline{\underline{A}}'$) and performing RCS as denoted as $\underline{\underline{A}}' \leftarrow \Psi_{RCS}(\underline{\underline{A}}')$ in Step (3). In case the new ACT has a cost larger than that of the old one, the newly derived ACT is simply disregarded, and the next greedy cycle will perform RCS based on the old ACT again. Such a greedy cycle is iteratively performed until a given number (100 in the simulation) of iterations of RCS have been done without getting a smaller cost at Step (4).

M-Trail Formation

This subsection introduces the basic idea of our m-trail formation mechanism, where Eq. (2.1) is used to evaluate $\underline{\underline{A}}$ in each greedy cycle in Fig. 2.11 such that the greedy cycle can possibly converge and yield a set of feasible m-trails with high quality.

Fig. 2.12 elaborates Step (2) of Fig. 2.11 through an example by considering a simple five-link topology. Initially, a 3-bit long alarm code is assigned to each link. The formation of the j^{th} m-trail has to take all the links e with $a_{e,[j]} = 1$ (which belong to L_j) as shown in Fig. 2.12(b-d). The ideal situation is that an $\underline{\underline{A}}$ with J bits yields exactly J link sets, which

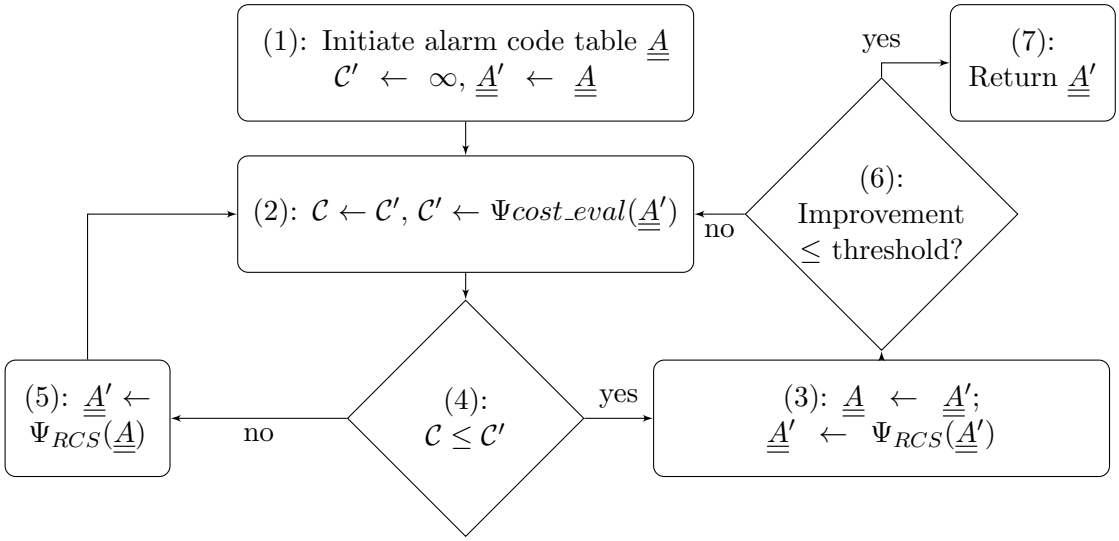


Figure 2.11: The flowchart for the proposed heuristic algorithm.

can form J valid m-trails. A link set forms an m-trail if all the links can be connected and traversed along a not necessarily simple path. In other words, the link set can have maximally two nodes with an odd nodal degree according to Euler's theorem. Checking this m-trail condition for a link set an m-trail can be done by a breadth-first search (BFS) algorithm in linear time. A link set could be far from interconnected and could even yield multiple isolated fragments. If a link set cannot be shaped into a valid m-trail (e.g. link set 3 of Fig. 2.12(d)), it will possibly be constructed as a union of multiple m-trails or cycles according to the following Lemma.

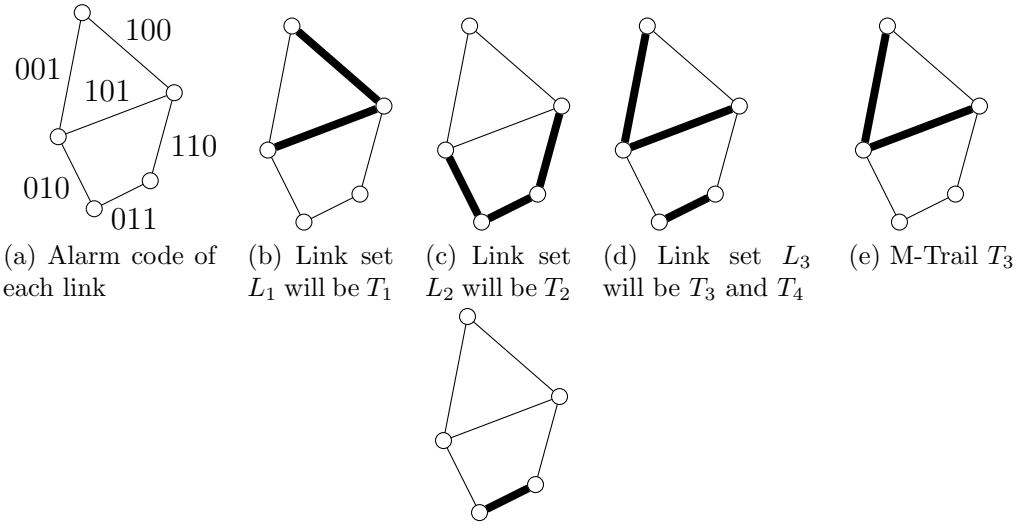
Lemma 4. *A connected graph can be efficiently decomposed into (1) a single cycle, if every node has an even nodal degree; or (2) a number of $\#odd(G)/2$ trails, where $\#odd(G)$ denotes the number of odd-degree nodes in the graph.*

Proof. The lemma is a consequence of Euler cycle and path theory. In both cases (1) and case (2), the cycle and the trails can be formed in linear time with Fleury's algorithm. □

The Lemma states that in case the i^{th} isolated fragment of link set of bit j (denoted as $C_{i,j}$) has more than two odd nodes (denoted by $\#odd(C_{ij})$), then it can be decomposed into the $\#odd(C_{ij})/2$ m-trails. This is also exemplified in Step (2.2) of Fig. 2.12.

In case at a specific bit position the links with "1" bit do not form a trail, we can always "separate" those links into several trails. And each trail is going to be a separate m-trail.

Random Code Swapping (RCS) The initial RCA may yield a unqualified result that contains many isolated fragments and a large number of odd-degree nodes. This subsection describes the proposed RCS mechanism for shaping the links of a link set into one or a number of m-trails while still meeting the overall UFL requirement. The key idea of the proposed RCS mechanism is the strong locality constraint (SLC) which governs the swapping mechanism in each link set. It means that the alarm code of a specific link in L_j can be swapped with that of a link not in L_j if all the other link sets are not affected due to the swapping. The necessary condition for meeting the SLC is that the alarm codes



(a) Alarm code of each link (b) Link set L_1 will be T_1 (c) Link set L_2 will be T_2 (d) Link set L_3 will be T_3 and T_4 (e) M-Trail T_3

(f) M-Trail T_4

Figure 2.12: An illustration of the cost evaluation method of an alarm code table of the 4 node line graph.

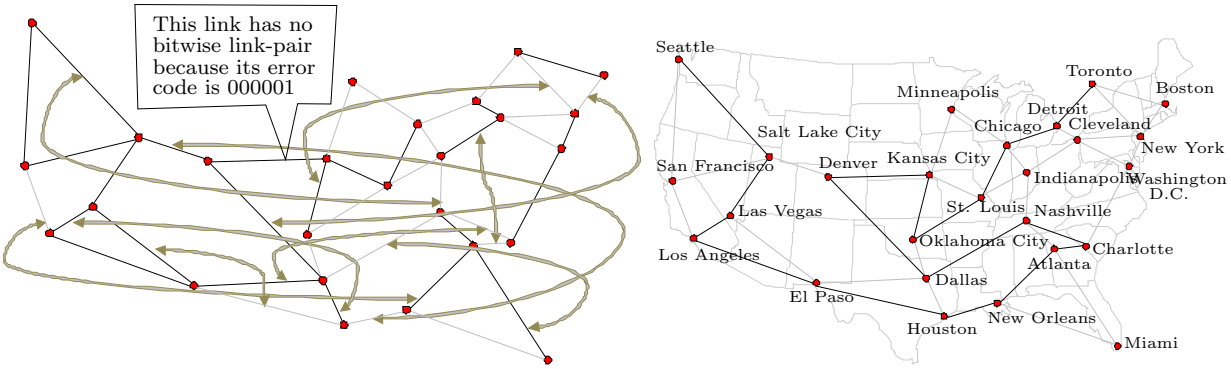
of two links in L_j are bitwise identical except for a single bit at position j. Such a code pair is called a *code pair of* L_j, and the two links corresponding to the code pair form a *bitwise link-pair of* L_j. For example, 1011011 and 1010011 form a code pair of L_4, and the corresponding links form a bitwise link-pair of L_4. Thus, swapping alarm codes of the two links meets the SLC due to the local influence on L_4. With the SLC, the RCS on a link set can be performed independently from the others. this mechanism allows easy implementation and provides high efficiency.

Note that for L_j, some links may not have a bitwise link-pair due to two reasons: (1) its code pair of L_j is all 0's, which does not correspond to any failure state. For example, 010000 is a code pair of 000000 of L_2, but there is not a link corresponding to the alarm code 000000. (2) The code pair of L_j of the link was not assigned to any link. In this case, the unassigned code can be freely used by the link without violating the SLC.

In summary, RCS is performed on each link set by randomly swapping alarm codes of all bitwise link-pairs of the link set, in order to help interconnecting isolated trail fragments and reducing the number of odd-degree nodes in the link set iteratively in each greedy cycle. The prototype of the proposed algorithm can be found in [53].

An Example on RCS Algorithm We provide an example here to show how RCS is performed. A 26-node network of US cities considered with 42 links. Initially with 6 bit long unique random codes were assigned to the links. Fig. 2.13(a) shows the link set assigned to the lowest bit (i.e., the 6^{th}). Except for the link between Denver and Kansas City that was assigned with an alarm code 0000001, all the other links either have a bitwise link-pair in L_6, or are don't-care links of L_6. Fig. 2.13(a) shows each link-pair at L_6 by an arrow. For example, (Atlanta, Charlotte) and (Indianapolis, Cleveland) are bitwise link-pairs of L_6. It can be easily seen that swapping the two links will interconnect two isolated fragments and reduce the number of odd-degree nodes by two, which leads to a saving of an m-trail. Similarly, swapping link (Salt Lake City, Denver) with link (Houston, New Orleans) will not increase the total cost of the ACT. While in the subsequent greedy cycle, swapping link (Las Vegas, El Paso) with link (El Paso, Houston) would further

reduce the number of m-trails through the RCS and thus possibly reduce the total cost. With more greedy steps on those link pairs and don't-care links of L_6, it is possible to form a single m-trail corresponding to the 6^{th} bit in the ACT while keeping all other link sets not modified. By iterating the greedy process for each bit in the ACT, the algorithm can guarantee to obtain an m-trail solution for each bit in the ACT. The solution quality will depend on the success rate threshold defined in Step (6) in Fig. 2.11. The effectiveness and efficiency of the proposed algorithm will be further demonstrated in the next section.



(a) (b)

Figure 2.13: Link set 6 (a) after random coding (b) after greedy random code swapping for link set 6 in the ACT.

Performance Evaluation of RCA-RCS

Extensive simulation on thousands of different random topologies was conducted to verify the proposed algorithm. Specifically, we (1) demonstrated the solution quality of the proposed algorithm by comparing to the ILP in [27]; and (2) investigate impacts of topology diversity on m-trail solutions.

Quality of Solution We investigated the quality of m-trail solutions generated by the proposed algorithm and its relation with the granted computation time. Also, comparison was made between the results by our algorithm and by the ILP in [27]. A server with 3GHz Intel Xeon CPU 5160 was used on two typical networks: SmallNet (10 nodes, 22 links) and ARPA2 (21 nodes, 25 links). Fig. 2.14 shows the quality of m-trail solutions in terms of the total cost defined in Eq. 1 versus the granted running time by taking cost ratio $\gamma = 5$. Each little data interval in Fig. 2.14 is 95% confidence interval around the mean of 30 experiments. The ILP solutions for the two networks under the same condition were also plotted for comparison.

With the RCS mechanism, the algorithm can achieve better results (i.e., smaller total cost) when longer computation time was granted in both networks. The simulation results confirmed our expectation. Interestingly, the proposed algorithm has generated even better solutions than the ILP, while the running time is shorter by several orders. This explicitly demonstrates the superiority of the proposed algorithm in terms of both solution quality and required running time against the ILP. Note that the ILP results in the example have nonzero gap-to-optimality of 4.17% in SmallNet and 20.41% in ARPA2, which cannot be erased only with dramatically increased running time. On the other hand, the proposed algorithm spent about 0.01 seconds and 1 seconds to achieve the

same total cost in the SmallNet and ARPA2, respectively, compared with 1,543 seconds and 9,573 seconds by the ILP.

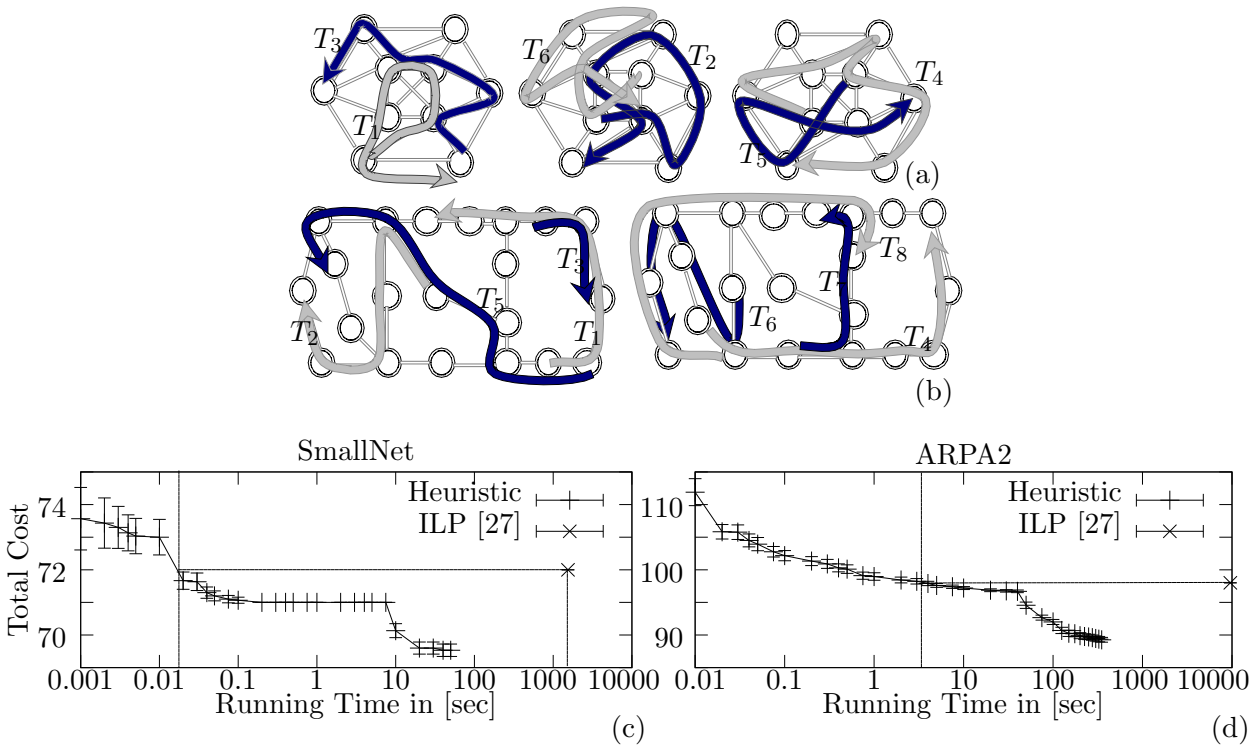
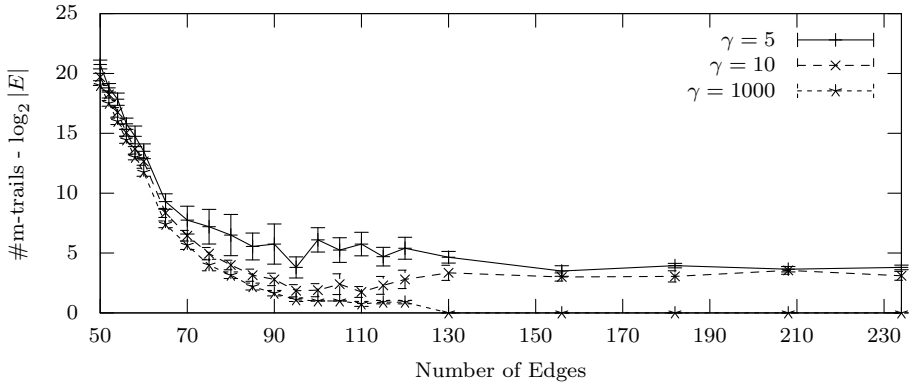


Figure 2.14: The best heuristic solutions for SmallNet with cost of 69 and ARPA2 with cost of 87. In comparisons the ILP resulted 72 with 4.17% gap-to-optimality for and SmallNet and 98 with 20.41% for ARPA2.

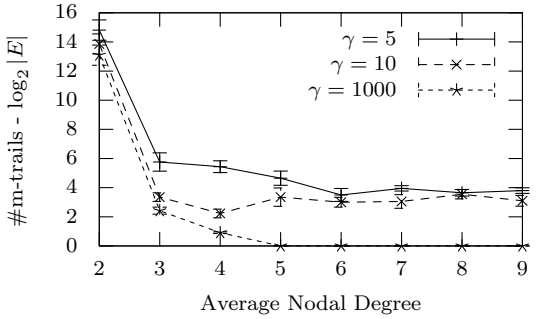
Topology Diversity on M-Trail Solutions This section demonstrates the impact on m-trail solutions due to topology diversity. A huge number of experiments on thousands of randomly generated topologies were conducted. Fig. 2.15 shows the minimal number of m-trails versus network density on topologies with 50 nodes. The network connectivity was increased starting from a backbone ring with 50 nodes and 50 links to a fully meshed topology with 50 nodes and 1,225 links, where one or a few links were randomly added to the topology for each data set. To make it statistically meaningful, every data interval in Fig. 2.15 is 95% confidence interval around the mean of 20 different topologies each obtained by randomly adding the same number of links to the backbone ring. We observed that the normalized length of the alarm code (i.e., from the length of alarm code we subtracted $\lceil \log_2 (|E|+1) \rceil$) dramatically goes down when we add 50 to 100 links, or increase the nodal degrees from 2 to 3. See Fig. 2.15(a) and Fig. 2.15(b), respectively. The length of the alarm code approaches the lower bound (i.e., $\lceil \log_2 (|E|+1) \rceil$) when γ is large enough. From Fig. 2.15(a) and (b), we have also observed that when γ is small, the confidence interval for each data is larger than in the case of larger γ. This indicates the fact that both monitoring cost and bandwidth cost are more sensitive to different amounts of degree-2 nodes in the network, possibly due to the interplay between the two objectives in the cost function of Eq. (2.1).

Fig. 2.15(c) shows the normalized cover ratio (i.e., the sum of cover length of all m-trails divided by $|E|$) versus average nodal degree. We have seen that the cover ratio

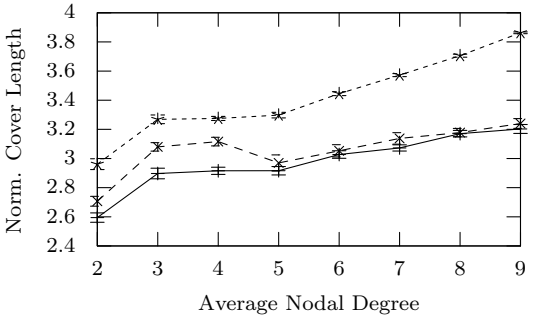
slightly increases when the average nodal degree is increased from 2 to 9 for all the three γ values. Particularly, the cases with $\gamma = 5$ and 10 have better suppressed the increase of cover length ratio, which demonstrates the effectiveness in the tradeoffs between the length of the alarm code and bandwidth cost by manipulating the cost ratio γ.



(a) Normalized length of alarm code versus number of links



(b) Normalized length of alarm code versus the average nodal degree

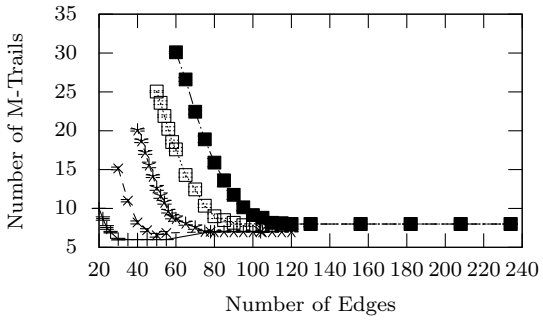


(c) Normalized cover ratio versus average nodal degree

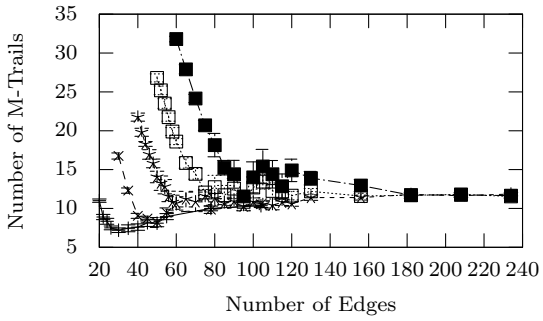
Figure 2.15: Simulation on random topologies with 50 nodes.

Fig. 2.16 shows the m-trail solutions with different numbers of nodes in the network topologies. Fig. 2.16(a) and (b) shows the length of the alarm code versus the number of links. Clearly, when $\gamma = 1000$, the length of the alarm code is only affected by the number of links when it is small, while quickly converges to $\lceil \log_2(|E|+1) \rceil$ when the number of added links is increasing, regardless the number of nodes in the topology. Moreover, the length of alarm code is always $\lceil \log_2(|E|+1) \rceil$ in case the network does not contain any node with a nodal degree 2 or smaller, which also verifies the observations. On the other hand, when γ is small, the convergence becomes slower, and the length of alarm code deviates more from $\lceil \log_2(|E|+1) \rceil$ as γ is reduced.

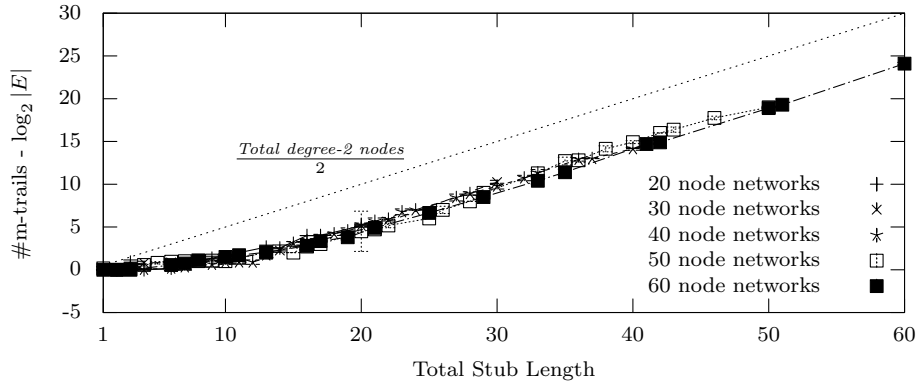
Fig. 2.16(c) shows the results of the experiment, on the relationship between normalized length of the alarm code and total number of degree-2 nodes in the topologies with 20, 30, 40, 50, and 60 nodes, altogether resulting 5,320 different random topologies. γ was set to 1,000. Each data point was obtained by averaging the results on 20 randomly generated topologies of the same total number of degree-2 nodes. We observed that all the topologies with different numbers of nodes require more m-trails for UFL as the number of degree-2 nodes grows, which meets our expectation. Interestingly, it is observed that all the topologies have to take similar normalized length of alarm code for UFL given the same number of degree-2 nodes. This indicates a serious impact on m-trail solutions due



(a) $\gamma = 1000$



(b) $\gamma = 5$



(c) Normalized length of alarm code versus total degree-2 nodes

Figure 2.16: The length of alarm code versus the number of added links for topologies of 20, 30, 40, 50, and 60 nodes.

to the number of degree-2 nodes in a topology. Based on these experiments the following relationship can be taken as a rule of thumb for approximating the normalized length of the alarm code in a random network topology when γ is very large:

$$b = \#\text{m-trail} \approx \lceil \log_2(|E|+1) \rceil + \frac{\#\text{degree-2 nodes}}{2} \tag{2.8}$$

Note that the above rule of thumb does not hold for every graph because the construction in Section 2.2.2 is a counterexample.

2.3 Unambiguous Failure Localization for Multiple Failures

2.3.1 Problem Definition

The alarm code of a multi-link SRLG is the *bitwise OR*, denoted by $\vee$, of the alarm codes of all links in the SRLG. This corresponds to the fact that a monitor alarms if the corresponding monitoring lightpath traverses through any link in the SRLG that is hit by the failure event. Thus, each SRLG z can be assigned with an alarm code as follows

$$A_z = \bigvee_{e \in \mathcal{Z}} A_e \quad ,$$

where $\bigvee$ denotes the *bitwise OR* operator. The *UFL constraint* in this general case requires every alarm code of each SRLG to be unique.

2.3.2 UFL for SRLG Failure with Bm-Trail

In general, an effective monitoring structure allocation method must satisfy the following two requirements, either in a single step or one after the other:

(R1): Every SRLG should be uniquely coded.

(R2): Each monitoring structure must be an eligible fragment of network topology in which a lightpath can travel along from the transmitter to the receiver.

Note that in addition to (R1) and (R2), there could be some other constraints due to specific user design premises, such as the length limitation due to the deployment of optical generators/retransmitters, the locations of monitoring nodes [35, 40], and use of working lightpaths (i.e., live connections) for failure state correlation [35, 42].

An integer linear program can be developed that satisfies both (R1) and (R2) in a single step [22, 27, 38]. In particular, [27] is the first study that suggested to using freely routed open-loop un-directed supervisory lightpaths (called m-trail) for single-link SRLG failure localization. All the three studies formulated the supervisory ligthpath allocation problem into Integer linear programs (ILPs), which is unfortunately subject to intolerably long computational time even in very small topologies. Thus people have turned to the design of heuristics in solving the problem. The previously reported solutions can be divided into two categories according to their design principles. The first one manipulates an accumulation mechanism such that (R2) is ensured at the beginning, while the goal of the heuristics is to satisfy (R1) [30, 35, 36]. In the second design category, (R1) is intrinsically ensured at the beginning while leaving (R2) as a goal [5].

The bm-trail formation problem is a structured variant of the combinatorial group testing (CGT) process [54, 55]. In [30] the problem was taken as *combinatorial group testing on graphs*. The idea of group testing dates back to World War II when millions of blood samples were analyzed to detect syphilis in US military. In order to reduce the number of tests it was suggested to pool the blood samples. From algorithmic aspects there are two significant differences between the tasks of pooling blood samples and monitoring of a group of links in a graph: (1) the blood samples can be pooled arbitrarily, while the monitored links must be connected and even in a valid shape, (2) the monitors are always pre-configured, and the probing is performed simultaneously without knowing the result of other tests (non adaptive CGT).

The primary goal of any CGT algorithm is to identify defective items among a given set of items through as few tests as possible. In this case the set of items are the links of a graph, the defective items are the failed links, and the tests are monitoring structures (e.g., bm-trails). The first general CGT method was given by Hwang and T. Sós [54], while the shortest real-world problem size non-adaptive CGT codes were developed by Epstein, Goodrich and Hirschberg [56].

In [30], with the help of non-adaptive combinatorial group testing (CGT) constructions, the authors conducted an indepth theoretical bound analysis on the problem, in which each link is assigned with multiple codes in a graph with at least $d+1$ disjoint spanning trees. Therefore, the construction in [30] can only be applied to very densely meshed topologies. For example, the network has to be as densely meshed as $(2d+2)$-connected in order to accommodate $d+1$ disjoint spanning trees, which results in $(d+1)\cdot j$ bits assigned to each link for achieving UFL of SRLGs with up to d links, where j is the CGT code length. Obviously, such a method of assigning each link $d+1$ CGT codes can fit well into theoretical analysis, but it can hardly be applied in most practical scenarios.

The studies in [35, 36] set their goal in minimizing the number of monitoring locations (MLs). For example, to localize failure of SRLG with up to 2 links (i.e., $d=2$), all the 3- and 4-connected subgraphs should be identified, and almost each subgraph needs an ML at an arbitrarily chosen node in the subgraph. With each ML determined, graph transformation is performed such that the MLs are merged into a supernode (denoted as m), and cycles are cumulatively added into the transformed graph one by one via Suurballe's [57] algorithm. To distinguish two SRLGs w_1 and w_2, a cycle must be disjoint from w_2 while passing m and l, where l is a link randomly selected from $w_1 \setminus w_2$. In the worst case this leads to $O(|SRLG|^2)$ of cycles to distinguish all the SRLGs, where $|SRLG|$ is the number of SRLGs considered in the network. Thus, the worst time complexity is $O(|SRLG|^2 \cdot |V|^2)$, where $|V|$ is the number of nodes in network, and the term $O(|V|^2)$ corresponds to the complexity of Suurballe's algorithm. The computation complexity becomes $O(|E|^{2d} \cdot |V|^2)$ if every multiple failure with up to d links should be localized, where $|E|$ is the number of links.

The approach taken in [5] (see also Section 2.2.7) is the first study following the second design principle, where the code uniqueness of each link (as defined in (R1)) is first guaranteed, while an algorithm was given for the formation of each monitoring structure in the context of m-trail. A superb performance was witnessed in [5] by employing random code assignment (RCA) and random code swapping (RCS) for localizing any link failure. In specific, the RCA algorithm forms the j-th m-trail by randomly swapping a link code with its *bitwise code pair* at the j-th position. For example, the codes "11010110" and "11000110" form a bitwise code pair at the 4-th position. Note that such a RCS algorithm

in [5] can only work when single-link failures are considered and it simply fails in presence of the code dependency among overlapped SRLGs, which is the most critical task to be addressed in this section.

This sections follows the second design category in order to take advantage of the extremely flexible structure of bm-trails in solving the problem.

2.3.3 The CGT-GCS Heuristic Approach for M-Trail Allocation

The proposed bm-trail allocation method follows the second design principle, where CGT codes generated by [56, 54] are assigned to each link to ensure (R1). After the random code assignment, (R2) is pursued by way of greedy code swapping (GCS). Although seemingly similar to that in [5], the proposed approach is much different in both stages of code generation and code swapping.

Fig. 2.17 is a flowchart that summarizes the proposed approach. In Step (1), the CGT code construction `GEN_CGT` generates a number of k $\overline{d}$-separable codes of a length J bits, denoted as $\underline{\underline{C}}$. The input parameter $|E|$ ensures that the code length J is the smallest such that $k \geq |E|$. Note that the property of $\overline{d}$-separability ensures uniqueness of the *bitwise OR* of up to d codes in set $\underline{\underline{C}}$, which is required in (R1). In Step (2), an alarm code table is formed by randomly selecting $|E|$ out of k codes from $\underline{\underline{C}}$, which are further assigned to all the links. The group of $|E|$ codes taken by the links is denoted as $\underline{\underline{A}}$, while the group of rest $k - |E|$ unassigned codes is denoted as code set $\underline{\underline{U}}$, where $\underline{\underline{U}} = \underline{\underline{C}} \setminus \underline{\underline{A}}$.

With a CGT code of length J at each link, the best situation is that each bit position of the link can lead to an m-trail, and in this case there are totally J m-trails corresponding to the code assignment. But this is not likely to happen due to the random assignment of the codes at the beginning. Our method solves the m-trail formation problem by GCS starting in Step (3), which ensures (R2).

In Step (3), each link is categorized with one of the four attributes (i.e., isolated, leaf, bridge, and detour) in each bit position according to $\underline{\underline{A}}$. Next in Step (4), code pair (a_e, C_x), where $a_e \in \underline{\underline{A}}$ and $C_x \in \underline{\underline{C}}$, is arbitrarily selected and checked in function `CostREval` one by one to see how much cost reduction can be achieved by possibly swapping each code pair. The code pair with the steepest cost reduction after swapping is kept (i.e., $\Delta_{\mathcal{M}}$). If $\Delta_{\mathcal{M}}$ is no less than γ and at least one bm-trail can be merged or removed, the two codes are swapped using function `SWP`, such that $\underline{\underline{A}}$ is updated accordingly in Step (6) and the program then goes back to Step (3). Otherwise, the program returns the best result (i.e., $\underline{\underline{A}}$ with the least bm-trails) and terminates.

Note that an eligible code swapping could be either a swapping between the codes both in $\underline{\underline{A}}$ (i.e., $C_x \in \underline{\underline{A}}$) or replacement of the link code with an unused one (i.e., $C_x \in \underline{\underline{U}}$). Steps (3), (4), (5), and (6) form a loop such that the largest cost reduction can be achieved in each iteration of code swapping.

Greedy Code Swapping (GCS)

To carry out m-trail formation, GCS is devised to greedily swap codes of two links such that the coverage constraint at each bit position can be satisfied while the resultant solution quality can be progressively improved according to the cost function Eq. (1). Such an iterative swapping process continues until a given condition is satisfied.

The cost reduction evaluation for each code swapping serves as an important building block in the proposed GCS mechanism, which guides the m-trail formation process at each link set. In swapping each code pair of two links, a set of regulations is necessary,

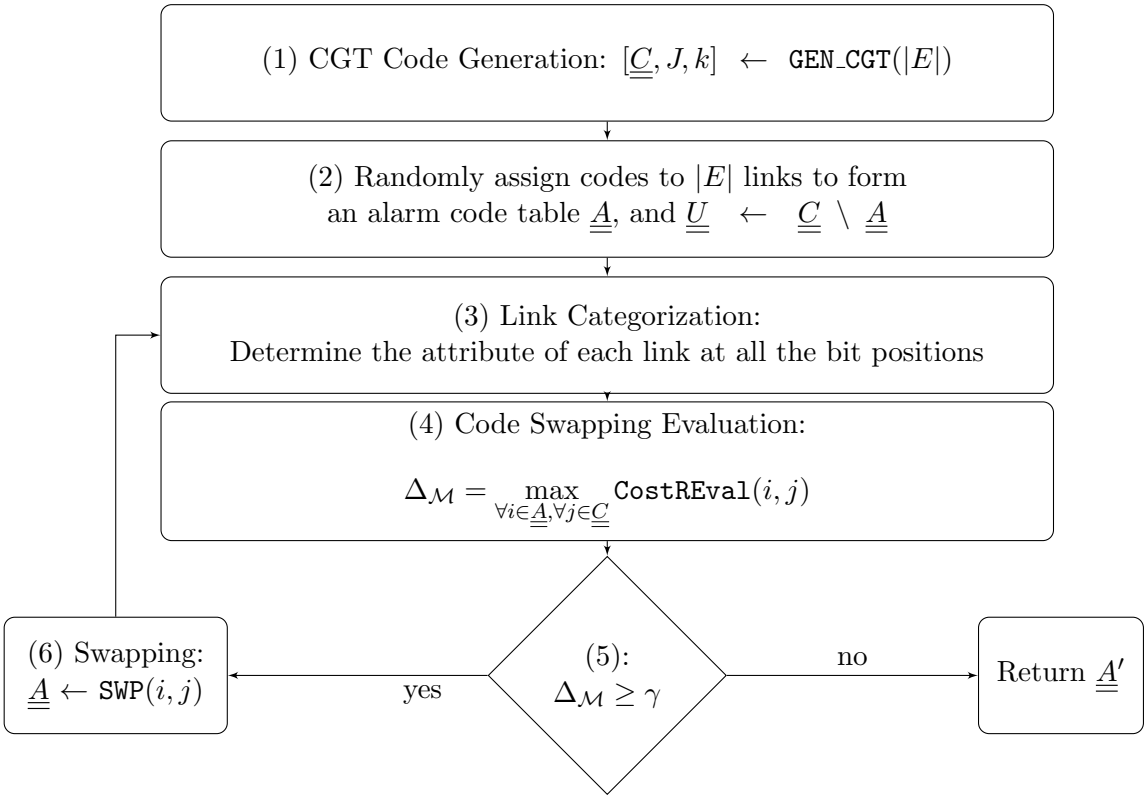


Figure 2.17: The flowchart for the proposed *CGT-GCS* heuristic algorithm.

and will be detailed in the following paragraphs.

The flowchart of the proposed GCS is given in Fig. 2.18, which provides all details of Step (4) in Fig. 2.17. At the beginning, the program picks up a code pair a_e and C_x as shown in Step (4.1), where a_e is a code assigned to link e while C_x is randomly selected from $\underline{\underline{C}}$, respectively. The cost reduction evaluation for a single swapping should be iterated on each bit position (or, each link set) affected by the swapping. The i-th bit position (or link set), L_i, is not affected by the swapping of a_e and C_x if the two codes have a common i-th bit, i.e., $a_{e,[i]} = C_x[i]$. If the swapping of a_e and C_x has an affection on the i-th link set, the heuristic goes to either Step (4.5) or (4.6), depending on whether $C_x \in \underline{\underline{A}}$ or $C_x \in \underline{\underline{U}}$, which is checked in Step (4.4). In the case $C_x \in \underline{\underline{U}}$, the function addBit$(e,i)$ is called if $C_x[i] = 1$ and $a_{e,[i]} = 0$; otherwise removeBit(i,e) is called if $C_x[i] = 0$ and $a_{e,[i]} = 1$. In the former case the i-th bit is flipped from "0" to "1", hence a link is added to L_i; while in the latter, the i-th bit is changed from "1" to "0", where a link is removed from L_i. If $C_x \in \underline{\underline{A}}$, let C_x be currently assigned to link f. The function add&removeBit(i,e,f) is called if $C_x[i] = 1$ and $a_{e,[i]} = 0$; otherwise the function add&removeBit(i,f,e) is called (i.e., $C_x[i] = 0$ and $a_{e,[i]} = 1$).

Before addBit(i,e), removeBit(i,e), add&removeBit(i,e,f) are introduced, the attributes of network links should be defined first, which facilitate high computational efficiency in the cost reduction proces for each link set.

The Attributes of Links A link set may contain one or multiple isolated fragments, which are called the *components* of the link set. Each link of link set L_j could be attributed into either one of the following four categories:

Isolated link is a link not connected to any other link of the link set. Identifying these

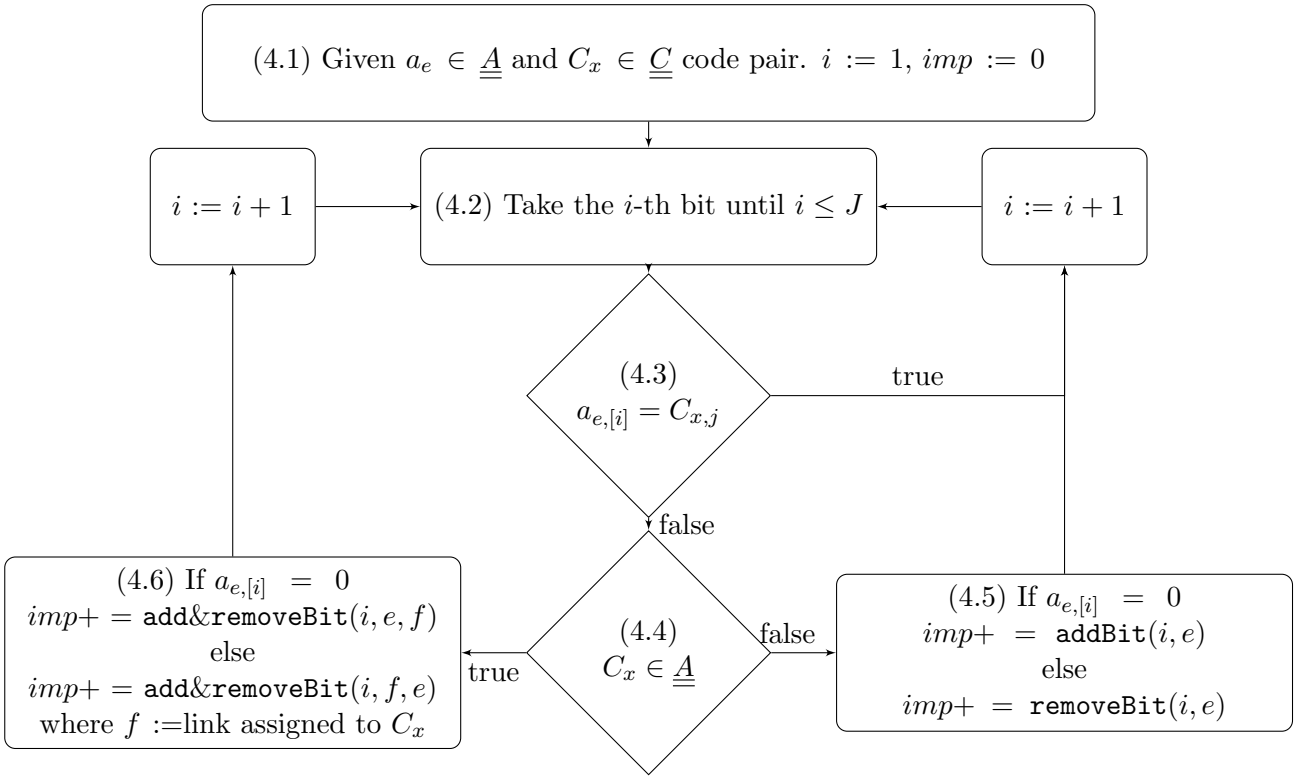


Figure 2.18: Cost reduction evaluation for each code swapping.

links is simple, since their both terminating nodes have degree 1. An example is given as (x, n) in Fig. 2.19.

Leaf link is a link with exactly one of its terminating nodes of nodal degree 1, as shown in link (c, r) and (u, z), etc., in Fig. 2.19.

Bridge link has both terminating nodes with a nodal degree larger than 1. Moreover, if the link is erased, then the component falls apart into two sub-components. To identify a bridge link, every 2-connected component must be identified first, which can be done in $O(|E|^2)$ time. For these links both terminating nodes of the link must belong to different 2-connected components. An example is given as (c, e) in Fig. 2.19.

Detour link is a part of a component,and removal of it does not tear the component apart. For these links both terminating nodes of the link must belong to the same 2-connected component. An example is given as (o, u) in Fig. 2.19.

Next, similar categorization is applied to each link set $\overline{L}_j$, where the isolated links, leaf links, bridge links, and detour links are identified.

`addBit`(i, e) returns the cost reduction in case $a_{e,[i]} = 0$ and $C_x[i] = 1$. In this case, because the i-th bit of the two link codes is changed from "0" to"1", the cover length of the resultant m-trail solution will be increased by 1, while the number of m-trails could be increased or reduced or unchanged according to the attribute of link e with respect to the link set L_i. Table 2.5 summarizes the link attribute categorization.

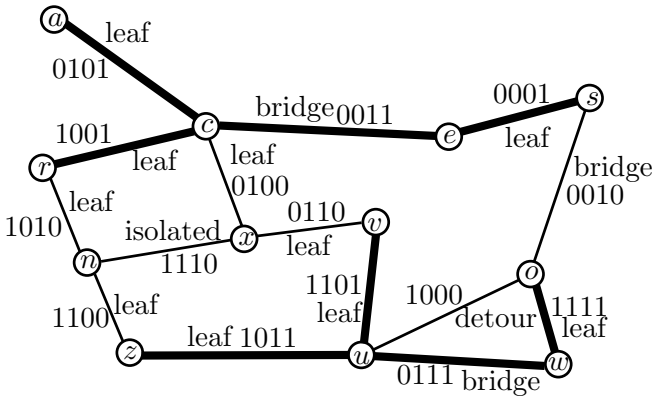


Figure 2.19: An example on link attribute categorization for fast cost reduction evaluation on code swapping.

Table 2.5: Table lookup of addBit(i, e)

Attribute of e for L_i	**# of m-trails**
isolated	increased by 1
leaf	unchanged
bridge	decreased by 1
detour	unchanged

`removeBit`(i, e) returns the cost reduction in case $a_{e,[i]} = 1$ and $C_x[i] = 0$. Because the i-th bit is changed from "1" to "0", the cover length is decreased by 1, while the number of bm-trails should be updated according to the attribute of e with respect to L_i. This is summarized in Table 2.6.

Table 2.6: Table Lookup of removeBit(i, e)

Attribute of e for L_i	**# of bm-trails**
isolated	decreased by 1
leaf	unchanged
bridge	increased by 1
detour	unchanged

`add&removeBit`(i, e, f) is for the cost reduction evaluation in the event that link e is added and another link f is removed from L_i. After the swapping the cover length is unchanged while the number of m-trails changes according to the attributes of both links. This is provided in Table 2.7.

In summary, the proposed GCS swaps a code pair with the steepest cost reduction larger than a threshold γ based on the proposed link attribute categorization and table lookup process, which greedily approaches to better performance according to Eq. (2.1). With GCS, very high computational efficiency can be achieved thanks to the constant time complexity in evaluating each code pair, which will be detailed in the next subsection. The prototype of the proposed algorithm can be found in [53].

Table 2.7: Table lookup of add&removeBit(i, e, f)

add e	remove f	
Attrib. of L_i	**Attrib. of L_i**	**# of bm-trails**
isolated	isolated	**unchanged**
	leaf	**increased by 1**
	bridge	**increased by 2**
	detour	**increased by 1**
leaf	isolated	**decreased by 1** if e and f are not adjacent links, otherwise **unchanged**
	leaf	**Either unchanged or increased by 1**, if e is connected to f. See link (r, n) and (r, c) on Fig. 2.19 as an example.
	bridge	**increased by 1**
	detour	**unchanged**
bridge	isolated	**decreased by 2** if e and f are disjoint, otherwise **increased by 1**
	leaf	**Either decreased by 1,** or unchanged if e is adjacent to f. See link (o, w) and (o, s) on Fig. 2.19 as an example.
	bridge	**unchanged**
	detour	**decreased by 1**
Detour	isolated	**decreased by 1**
	leaf	**unchanged**
	bridge	Either **decreased by 1** or **unchanged** if e reconnects the detached sub-components. See links (o, u) and (u, w) in Fig. 2.19 as an example. Lemma 5 provides a method to determine a bridge.
	detour	**unchanged**

Computational Complexity Analysis

The cost reduction evaluation is performed in each code swapping, which dominates the computational complexity of the heuristic algorithm. Next we prove three claims needed for a lemma that describes the computational complexity of the cost reduction evaluation process for a single code swapping.

Lemma 5. *The complexity of* `CostREval`(i, j) *is* $O(1)$.

Proof. In the table lookup process for each code swapping, it can be intuitively verified that every entity in the table can be performed in constant time.

Note that it is not obvious that the execution of `add&removeBit`(i, e, f) with e and f as a detour and bridge link, respectively, only takes constant time complexity. It is achieved via a pre-calculation process performed beside the link attribute categorization in Step (2).

For link set L_j, we need to determine the relationship between any node and a bridge link of $a_j = 1$, which can be done in constant time. For example in Fig. 2.19(a), (c, e) is a bridge link of L_j, and removal of it separates L_j into two isolated components that form two m-walks with nodes $\{a, c, r\}$ and $\{e, s\}$, respectively. Such a function can be

implemented by storing the *reach* and *leave* order of each node in the DFS algorithm. As shown in Fig. 2.20 as an example, the reach order of the DFS is written on the top of nodes, while the leave order is below the nodes. Let I_R and I_R denote the largest reach and the smallest leave order indices of the bridge. Every node with a reach and leave index at least I_R and at most I_L belongs to one side of bridge. As exemplified in Fig. 2.20, we have $I_R = 2$ and $I_L = 8$, thus $\{a, r, c\}$ are in one sub-component. □

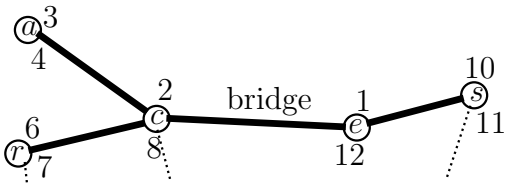


Figure 2.20: Leave and reach order of the DFS algorithm. The links of L_j are drawn with thick lines.

Lemma 6. *The complexity of Step (2) is* $O(|E| \log |E|)$.

Proof. Clearly, a DFS function for detecting the components in each link set is in $O(|E|)$ time complexity. Since each bridge link connects to two 2-connected components, to identify a bridge link, we must identify the corresponding two 2-connected components first, which can be done in $O(|E|)$ time complexity[58]. Also, such a check needs to go through each bit position, which multiplies the complexity by a factor of $\log |E|$. Thus, the lemma is proved. □

Lemma 7. *The complexity of Step (3) is* $O(|E|^2 \log |E|)$.

Proof. For each code and link pair, the proposed method can evaluate the possible cost reduction with a constant time ($O(1)$) according to Lemma 5. Since the overall time complexity is $O(|E| \cdot \#codes \cdot J)$, and since $\#codes = O(|E|)$, we have the worst case complexity of Step (3) as $O(|E|^2 \log |E|)$. □

Lemma 8. *The computational complexity of a cost reduction evaluation process for a single swapping is* $O(|E|^2 \log |E|)$.

Proof. It is a direct consequence of the above three lemmas. □

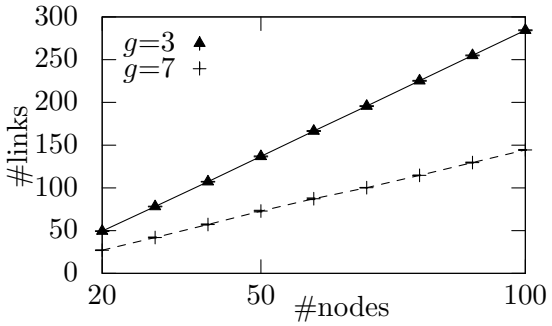
It is clear that the number of isolated components (or m-trails) in the initial random code assignment for each bit position cannot be more than $|V|/2$. This is because each isolated component consists of at least a single edge and two nodes, where $|V|$ is the number of nodes in the network. After each loop (defined in Steps (3), (4), (5), and (6) of Fig. 3), at least one m-trail is determined and erased from the link set; thus, the maximum number of code swappings should be upper bounded by $\frac{|V|}{2} \cdot J$, where J is the code length (in bits). Note that J is in the order of $O(d \cdot \log |E|)$ according to the CGT construction. By considering the complexity of each code swapping as $O(|E|^2 \log |E|)$, the overall worst case complexity in the proposed method is $O(|V||E|^2 \cdot d \cdot \log^2 |E|)$. Compared with the scheme in [35, 36] with a complexity of $O(|E|^{2d} \cdot |V|^2)$, the proposed approach can achieve much better efficiency.

Performance Evaluation of CGT-GCS

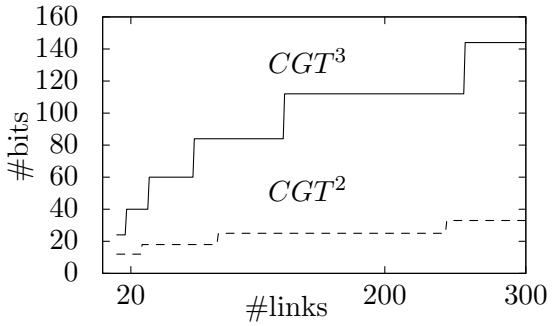
Simulations on hundreds of randomly generated planar 2-connected network topologies were conducted. The network topologies were generated with `lgf_gen`, a random graph generator of LEMON [59], which randomly generates realistic planar 2-connected networks. The networks are classified according to the girth of the graph, denoted by g, which is the length of a shortest cycle contained in the graph. Clearly, a smaller value of g yields a more densely meshed topology. Fig. 2.21(a) shows the statistics of the randomly generated topologies. See also Fig. 3.5 for example network topologies with different girth. It is found that the average nodal degree is 3.0 for dense networks ($g = 7$) and 4.0 for sparse networks ($g = 4$).

In the simulation, the proposed scheme is denoted as $CGT\text{-}GCS^1$, $CGT\text{-}GCS^2$ and $CGT\text{-}GCS^3$ for failure localization of SRLGs with up to 1, 2, and 3 links, respectively, where the CGT codes based on $\overline{d}$-separable constructions with $d = 1$, $d = 2$, and $d = 3$ are employed. CA^1 and CA^2 corresponds to the method that each bi-directional m-trail is allocated one after the other to distinguish each pair of SRLGs using any Dijkstra's algorithm based scheme, such that UFL for single-link SRLGs and for both single- and double-link SRLGs can be achieved, respectively. The method is generic and has been considered in a number of previously reported studies [35, 60, 26]. We have also implemented the construction in [30] that used disjoint spanning trees, denoted as $DSTC$. The construction provides an upper bound for $(d+1)$-connected topologies, but is invalid for topologies with any node of a smaller nodal degree than $(d+1)$. The upper bound is given by $(d+1)\ldots J$, where J is the length of the CGT codes employed.

Fig. 2.21(b) shows the lengths of CGT codes (i.e., J) versus the number of links $|E|$ of the corresponding topology by the CGT code generator `GEN_CGT` in [56], where the scenarios with $d = 2$ are presented. It is intuitive that when SRLGs with more links are considered, longer CGT codes are required for each link.



(a) Statistics of the random topologies. The dense networks have girth $g = 3$, while for the sparse networks $g = 7$.



(b) The length of CGT code (in bits) versus the number of links as an input to the CGT code generator `GEN_CGT` in [56]

Figure 2.21: Statistics on the input data.

The performance metrics employed in the comparison of the six schemes are the minimum number of m-trails required for achieve UFL and the running time. Both metrics are examined with respect to different network sizes (i.e., the number of nodes) and topology densities (i.e., g values), which will be presented in the following two subsections. The simulation has been done on over 800 randomly generated topologies, and each data was obtained by averaging the results from 10 different topologies with a specific g value and

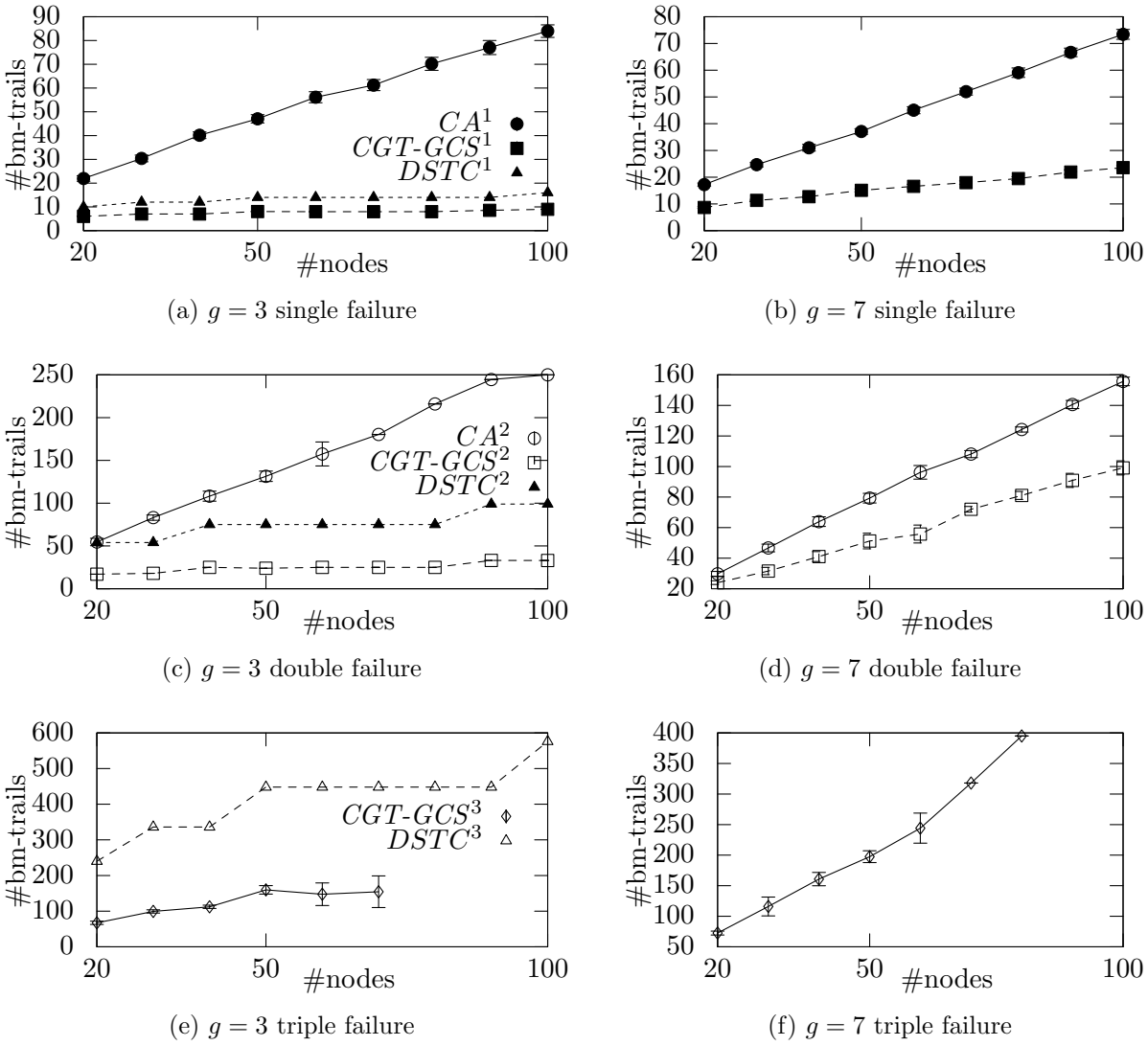
number of nodes. A bar for each data in the charts is available for showing the range of data we obtained.

Number of M-Trails versus Network Size The performance in terms of the minimum number of m-trails is first investigated, and the results are shown in Fig. 2.22. First, we find that the number of m-trails increases when the network size grows, which is observed in all the cases. It clearly shows that the proposed approach achieve much better scalability, where $CGT\text{-}GCS^1$, CA^1, and CA^2 have achieved far worse performance than that by $CGT\text{-}GCS^1$ and $CGT\text{-}GCS^2$, respectively, in both types of network topologies.

The superior performance of the proposed approach in minimizing the number of m-trails can be explained in two folds. First, the m-trails have the most flexible routing structure that can fully explore the solution space. This serves as a critical factor in overcome the vicious effect of topology diversity. It can be attested that our scheme has less advantage against the other two counterparts when network is sparsely connected (i.e., $g = 7$), because there are less alternatives in allocating the m-trails. Second, because CA^1 and CA^2 have each monitoring lightpath sequentially allocated into the network using an shortest path routing algorithm, it lacks intelligence in exploring the design space and network topology diversity. Third, $DSTC$ [30] tries to ensure the code uniqueness of each SRLG by useing $d + 1$ disjoint spanning trees, which is strongly limited by the topology connectivity. It is clearly shown that the construction can only yield valid solution in very densely meshed topologies, while failed in most of the sparse topologies considered in the simulation.

Running Time Fig. 2.23 shows the running time for obtaining the data in Fig. 2.22. It is observed that the proposed approach achieves much better computational efficiency, although CA^2 can achieve performance closer to $CGT\text{-}GCS^2$ for sparser network topologies. Also, $CGT\text{-}GCS^3$ takes much longer time than $CGT\text{-}GCS^2$ and any other case due to a much longer CGT code with $d = 3$, as shown in Fig. 2.21(b). Recall that the cost reduction evaluation in each code swapping has to go through all the link sets that are affected by the code swapping. Thus, the longer CGT code of each link yields an immediate increase of running time in obtaining an m-trail solution.

Note that we tried to implement CA^3 but failed due to the extremely long computational time required for each data in the selected topologies. This clearly demonstrates superior scalability of the proposed approach which can achieve better capability in handling a huge amount of SRLGs in densely meshed networks compared with its counterparts.



(a) $g = 3$ single failure

(b) $g = 7$ single failure

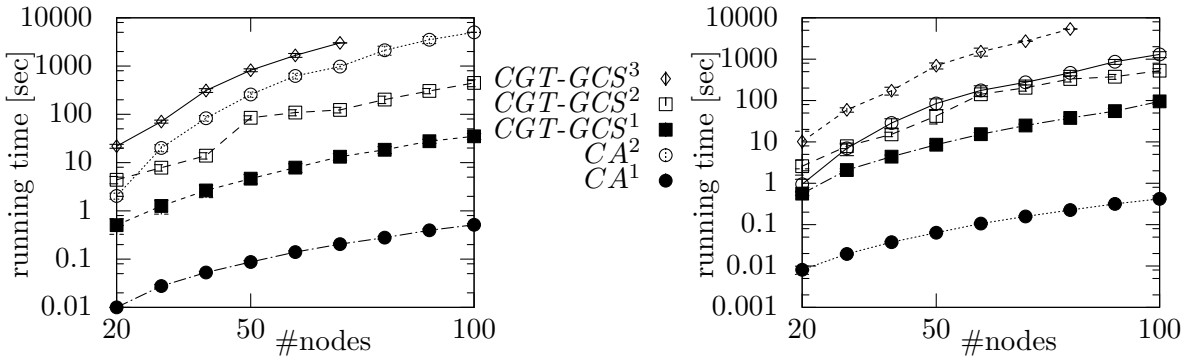
(c) $g = 3$ double failure

(d) $g = 7$ double failure

(e) $g = 3$ triple failure

(f) $g = 7$ triple failure

Figure 2.22: The number of bm-trails versus the number of nodes.



(a) Dense networks

(b) Sparse networks

Figure 2.23: The running time versus the number of nodes.

Chapter 3

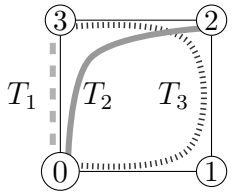
Distributed Single Failure Localization in All-Optical Mesh Networks

3.1 Introduction

Unambiguous Failure Localization (UFL) is defined in all-optical mesh WDM networks where any link failure can be precisely and instantly identified via monitoring a set of supervisory lightpaths. Such capability in the network optical layer is highly desired in order to meet the stringent requirement on service continuity and to support various failure-dependent restoration mechanisms [61, 62, 63].

Several UFL schemes have been proposed in Chapter 2. Using m-trails with bi-directional lightpaths in all-optical WDM networks has been proposed [30, 6, 64, 65]; and all these studies set the destination node of each m-trail as the only node that can detect the status of the m-trail. This results in a fact that alarm dissemination is needed upon a failure event mostly via flooding and dedicated failure notification based on network layer signaling, such that the alarm code can be formed at a remote site or any network node in order to localize the failure. Obviously, using electronic signaling in alarm bit collection incurs additional control complexity, operational overhead, and lower robustness of the system. The dependency on the upper layer signaling mechanism implies that these approaches leave the optical domain, which may cause problems. Such an issue was considered in [35], which aimed to minimize the number of monitoring locations (MLs) in order to reduce the alarm dissemination. Nonetheless, the research in [35] can hardly be applicable to the scenario where all the nodes are required to perform UFL in the optical domain, and may yield solutions with multiple MLs in sparse networks, and in this case the MLs have to exchange their alarm bits via control plane signaling. Most importantly, the approach in [35] pays no attention to the number of required supervisory lightpaths which affects seriously the number of transmitters and the total cover length.

Motivated by the fact that UFL should be carried out completely in the optical domain and be free from any control plane signaling effort, this section investigates an advantageous scenario of all-optical failure localization using bi-directional m-trails. Here we define *Local UFL* (L-UFL) at a node if the node can individually perform UFL based on locally available on-off status of the traversing m-trails; and *Network-Wide Local UFL* (NL-UFL) in the network if every node is L-UFL capable. By assuming that any node



(a) Topology and m-trails

SRLG	T_1	T_2	T_3
$(0,1)$	0	0	1
$(1,2)$	0	0	1
$(2,3)$	0	1	1
$(3,0)$	1	1	0

(b) Alarm code table (ACT) at node 0

Figure 3.1: Local-unambiguous failure localization (L-UFL) based on m-trails. Node 0 is the monitoring location. The cover length is $\|\mathcal{T}\| = 6$.

along an m-trail can obtain the on-off status of the m-trail via optical signal tapping, all the nodes traversed by the m-trail can share the on-off status of the m-trail. The proposed NL-UFL m-trail allocation problem is to find a set of m-trails such that every node can perform L-UFL based on the traversing m-trails.

The section first introduces the NL-UFL framework along with its possible application scenarios, and defines the NL-UFL m-trail allocation problem. To gain an understanding of the problem, we conduct bound analysis by proving optimal solutions in terms of minimal cover length in a number of special graphs, such as lines, stars, and complete graphs, as well as the derivation of lower bounds in general graphs. Inspired by the optimal solution for complete graphs, we develop a novel heuristic algorithm for solving the NWL-UFL m-trail allocation problem based on random spanning tree assignment (RSTA) and greedy link swapping (GLS). Extensive simulation on thousands of randomly generated topologies is conducted to verify the proposed heuristic approach and examine the derived lower bounds. The impact of network diversity is also investigated.

3.2 Problem Definition

3.2.1 Local Unambiguous Failure Localization (L-UFL)

L-UFL is an advanced application of m-trail deployment, where the UFL constraint is stricter while the shape constraint is the same as for UFL described in Section 2.1.2. A node is said to be *L-UFL capable* if and only if the node can perform UFL by inspecting the on-off status of the m-trails locally available. An m-trail is said to be *locally available* at node n if it terminates in n. In the L-UFL problem the goal is to find a node with L-UFL. This node is called *monitoring location (ML)*.

In many cases none of the nodes can serve as an ML, and thus the goal is to find a smallest set of nodes that altogether have sufficient information for UFL. As a consequence of Theorem 8, any node can be the single ML in a 2-connected network to achieve L-UFL with bm-trails for single link failures. The bandwidth cost is also considered in the L-UFL problem, with significantly smaller weight compared to the number of MLs.

3.2.2 An L-UFL Example

Fig. 3.1 shows an example of L-UFL m-trail solution on the network of 2.1(a) for localizing any single link failure. Node 0 is chosen for monitoring location, and the local ACT at node 0 is shown in Fig. 3.1(b). Here, every m-trail should terminate in node 0.

3.2.3 State of the Art on L-UFL

The study in [35] set its goal to minimize the number of MLs, which is determined by analyzing the connectivity among the 2- and 3-edge-connected components in the topology. With each ML determined, a graph transformation is performed such that the MLs are merged into a supernode (denoted by m), and cycles are cumulatively added into the transformed graph one by one via Suurballe's algorithm. The method was extended for sparse SRLG in [36].

3.2.4 Network-Wide L-UFL

In Network-wide L-UFL (NL-UFL) every node has to be L-UFL capable. To achieve this the L-UFL condition is slightly relaxed by assuming that any node along a monitoring lightpath can obtain the on-off status of the lightpath via optical signal tapping. Thus, all the nodes traversed by the lightpath can share the on-off status of the m-trail. It means a node can locally inspect the on-off status of every traversing monitoring lightpath.

The task is to allocate monitoring lightpaths with minimum total cover length, such that the shape constraint is fulfilled (see also Section 2.1.2), and the NL-UFL constraint requires unique alarm code for each SRLG at each node. The alarm code at node $v \in V$ for SRLG $z \in \mathcal{Z}$ is denoted by a_z^v, where $a_z^v = [a_{z,[1]}^v, a_{z,[2]}^v, \ldots, a_{z,[b^v]}^v]$ is a binary vector, b^v is the total number of m-trails traversing node v, and the jth bit of a_z^v, denoted by $a_{z,[j]}^v$, is 0 if the j-th m-trail traversing node v is operating after failure z, and 1 otherwise.

Thus for each node v we define an ACT, denoted as $\underline{\underline{A}}^v$, which is a $|\mathcal{Z}| \times b^v$ size matrix with each row as a_z^v, $\forall z \in \mathcal{Z}$. Obviously $b \geq b^v$ for each $v \in V$, as it is a sub-matrix of the global ACT. When any failure occurs and interrupts one or a number of m-trails, node v will obtain a nonzero alarm code which uniquely identifies the failed SRLG.

The target function of the NL-UFL problem is the total cover length, denoted by $\|\mathcal{T}\|$. It is because NL-UFL does not rely on any upper layer signaling effort for alarm code dissemination and, thus the length of alarm code is not as critical as that in UFL framework.

The following theorem indicates the feasibility of the approach in any 2-connected graph. The theorem demonstrates that an m-trail solution for single-link NL-UFL can always be obtained in a connected graph.

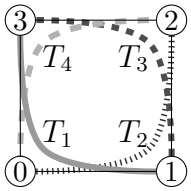
Theorem 8. *Given a connected graph, an m-trail solution for NL-UFL can always be found.*

Proof. The necessity of the connectivity of the graph is trivial. For the sufficiency part, we need to show that every node v_s can achieve L-UFL. Formally, we have to prove that the failure of an arbitrary link (v_i, v_j) can be unambiguously detected at v_s using a suitable collection of m-trails.

Indeed, after possibly renumbering the vertices we may assume, that v_i is not farther form v_s than v_j. Then let T_1 be a (possibly empty) shortest path from s to v_i. Note that T_1 does not contain the link (v_i, v_j). Also, let $T_2 = T_1 \cup \{(v_i, v_j)\}$. Then we have that (v_i, v_j) is faulty iff T_2 gives an alarm signal and T_1 does not. These events are obviously visible at any v_s and can be repeated for every link (v_i, v_j). □

3.2.5 An NL-UFL Example

Fig. 3.2(a) shows an example of NL-UFL for any single-link SRLG using four m-trails in a topology with 4 nodes. Fig. 3.2(b) shows the routes of the four m-trails T_1, T_2, T_3, and



(a) M-trails

Link	T_1	T_2	T_3	T_4
$(0,1)$	1	1	0	0
$(1,2)$	0	1	1	0
$(2,3)$	0	0	1	1
$(3,0)$	1	0	0	1

(b) Route table

Link	T_1	T_2	T_3
$(0,1)$	1	1	0
$(1,2)$	0	1	1
$(2,3)$	0	0	1
$(3,0)$	1	0	0

(c) ACT at node 1

Figure 3.2: Network-wide Local UFL (NL-UFL) via m-trails. The cover length is $\|\mathcal{T}\| = 8$.

T_4. In our case, each of the four nodes can achieve single-link L-UFL by inspecting the locally available on-off status of the traversing m-trails. For example, node 1 maintains an ACT as shown in Fig. 3.2(c), where the on-off status of T_1, T_2, and T_3 form an alarm code of three bits which uniquely maps to each possible link failure event. If node 1 finds that T_1 and T_2 become suddenly off while T_3 is still on, link $(0,1)$ is considered down and can be localized as defined in the first row of the ACT; if T_1 and T_2 are on while T_3 is off (as shown in the third row of the ACT), link $(2,3)$ is considered down. Similarly, node 0, 2, and 3 can perform UFL by maintaining their ACTs, each of which keeps the mapping between all the considered failure states and the on-off status of the traversing m-trails.

3.3 Bounds On Bandwidth Cost

This section presents our bound analysis for cover length in the proposed NL-UFL m-trail problem. We will first look into a lower bound on general connected graphs, followed by the optimal solutions for a number of special graph topologies: line, star and complete graphs.

3.3.1 Lower Bound for General Graphs

Theorem 9. *Let $\mathcal{T} = T_1, \ldots, T_b$ be a valid m-trail solution for NL-UFL. We have*

$$\|\mathcal{T}\| \geq \sum_{j=1}^{|V|} \sum_{\mu=1}^{|V|} \frac{\mu - 1}{\mu} \left(\lceil \log_2 \delta_j(\mu - 1) \rceil - \lceil \log_2 \delta_j(\mu) \rceil \right) \tag{3.1}$$

where $\delta_j(\mu)$ denotes the number of links whose shortest distance from node v_j is μ.

Proof. Let the *reputation* of T_i be denoted by $r(T_i)$, which represents the number of nodes along m-trail T_i that are aware of the on-off status of T_i. A trivial upper bound on $r(T_i)$ is $|T_i| + 1$ if T_i is loopless; or formally

$$r(T_i) \leq |T_i| + 1. \tag{3.2}$$

Let us define a matrix Ω with b columns and n rows, where

$$\omega_{i,j} = \begin{cases} \frac{|T_i|}{r(T_i)} & \text{the } i\text{-th m-trail traverses node } v_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The length of T_i can be expressed as

$$\sum_{j=1}^{|V|} \omega_{i,j} = \sum_{j=1}^{|V|} \frac{|T_i|}{r(T_i)} = |T_i|. \tag{3.4}$$

Thus we have

$$\sum_{i=1}^{b}\sum_{j=1}^{|V|}\omega_{i,j} = \sum_{i=1}^{b}|T_i| = \|\mathcal{T}\| \tag{3.5}$$

As a lower bound on $\omega_{i,j}$ according to Eq. 3.2, we have

$$\omega_{i,j} = \frac{|T_i|}{r(T_i)} \geq \frac{|T_i|}{|T_i|+1} \geq \frac{1}{2} \tag{3.6}$$

Let us denote the set of m-trails passing through node v_j by $\mathcal{T}^j$. Next, we give a lower bound on $\sum_{i=1}^{b}\omega_{i,j}$ for each node v_j, which is denoted by ω_j, formally

$$\omega_j = \sum_{i=1}^{b}\omega_{i,j} = \sum_{T_i\in\mathcal{T}^j}\frac{|T_i|}{r(T_i)} \geq \sum_{T_i\in\mathcal{T}^j}\frac{|T_i|}{|T_i|+1} \tag{3.7}$$

where $T_i \in \mathcal{T}^j$ means that m-trail T_i passes through node v_j. Note that the cover length is the sum of ω_j for all nodes, formally

$$\|\mathcal{T}\| = \sum_{j=1}^{|V|}\omega_j, \tag{3.8}$$

Let us denote the distance between node v_j and link $e \in E$ by $\delta(v_j, e)$, which equals to the length of the shortest path between node v_j and the closest adjacent node of e plus 1. Without loss of generality, the distance can be simply measured in hops. Let $\delta_j(k)$ denote the number of links whose distance from node v_j is at least k. For example, $\delta_j(1) = |E|$ as every link is at least 0 hop from the node, and $\delta_j(2) = |E| - \Delta_j$, where Δ_j is the nodal degree of j. Let n_j denote the maximum δ distance from node v_j, thus $\delta_j(n_j+1) = 0$ and $\delta_j(n_j) > 0$. Note that such n_j always exist because there are no links with distance $n-1$. Note that $\delta_j(\mu)$ is a monotonically non-increasing function of μ (i.e. $\delta_j(1) \geq \ldots \geq \delta_j(n_j) \geq 1$). To localize any single failure at node v_j among links whose distance from v_j is at least $\mu - 1$, there must be $\lceil \log_2 \delta_j(\mu) \rceil$ m-trails with at least a distance of μ.

From $\mathcal{T}^j$ we select $\lceil \log_2 \delta_j(n_j) \rceil$ m-trails whose length is at least n_j. By Eq. (3.7) they contribute to ω_j at least

$$\frac{n_j - 1}{n_j}\lceil \log_2 \delta_j(n_j - 1) \rceil = \frac{n_j - 1}{n_j}(\lceil \log_2 \delta_j(n_j - 1) \rceil - \lceil \log_2 \delta_j(n_j) \rceil).$$

In a similar manner, from the remaining m-trails in $\mathcal{T}^j$ we may select as many as $\lceil \log_2 \delta_j(n_j - 2) \rceil - \lceil \log_2 \delta_j(n_j - 1) \rceil$ m-trails whose lengths are at least $n_j - 2$. Their contribution to ω_j is at least

$$\frac{n_j - 2}{n_j - 1}(\lceil \log_2 \delta_j(n_j - 2) \rceil - \lceil \log_2 \delta_j(n_j - 1) \rceil),$$

by Eq. (3.7) again. Continuing in this way for $\mu = n_j, \ldots, 1$ we obtain a collection of m-trails in $\mathcal{T}^j$ such that the following inequality follows

$$\omega_j \geq \sum_{\mu=1}^{n_j}\frac{\mu - 1}{\mu}\left(\lceil \log_2 \delta_j(\mu - 1) \rceil - \lceil \log_2 \delta_j(\mu) \rceil\right) \tag{3.9}$$

By summing up ω_j for every node according to Eq. (3.8) we get a lower bound on cover length in Eq. (3.1). □

Theorem 9 can be well applied when the average nodal degree is small (e.g., less than 6). We show in Section 3.4.3 that Theorem 9 can provide a lower bound in a form $\|\mathcal{T}\| \gtrsim \xi|V|\log_2(|E|+1)$ where ξ is somewhere in $[0.85, 0.95]$ for random networks with average nodal degree less than 6.

To further improve the above theorem, next we introduce a related Combinatorial Group Testing (CGT) problem. We will first consider the lower bound on this generalized version of Combinatorial Group Testing (CGT) and then apply them to the NL-UFL requirement at each node, which will give us a lower bound on the coverlength for general graphs. The key idea is to define a special cost function for the m-trails at each node such that the lower bound to meet the NL-UFL problem of each node can be summed up to get a lower bound on the total coverlength.

3.3.2 General Lower Bound for CGT

Let us consider a non-adaptive CGT problem where the goal is to find one faulty item among a set of items with group tests, where each group test is on a set of items and has two an outcomes: the test contains a faulty item or not. Note that the NL-UFL problem at each node n is a special version of CGT, where the tests are the m-trails passing through n, and the items are the links. We have two additional constraints:

- The links must form a path in the topology, and
- Each of the links must have a non-zero code.

It is clear that a valid NL-UFL solution at node n is a valid CGT solution over links E.

Next, let us formalize the CGT problem with a cost function on each test. The cost of test T_i depends on its size according to a given cost function $\omega()$. The input of the CGT problem is a set of items denoted by $E = \{e_1, \ldots, e_m\}$ and a cost function ω, where $m = |E|$ is the number of items. The goal is to establish a set of b group tests, denoted by $T_1, \ldots, T_b$, where each group test consists of a set of items, such that a single faulty item can be unambiguously identified according to the outcomes of the group tests. It is also called *separating* test collection. Each test has a cost defined as follows

Definition 1. *The cost of test T_i with $t_i = |T_i|$ is $\omega(t_i)$, where function ω has the following properties:*

(i) $\omega(1) = 1$, *meaning that test with one element has a unit cost.*

(ii) $\omega(x+1) \geq \omega(x)$ *for every positive integer* $1 \leq x \leq m-1$. *Testing a larger group cannot decrease the cost.*

(iii) $\frac{\omega(x)}{x} \geq \frac{\omega(x+1)}{x+1}$ *whenever* $1 \leq x < m$.

The goal is to identify the faulty item with minimum cost:

$$\text{Minimize } \Omega = \sum_{i=1}^{b} \omega(t_i) \tag{3.10}$$

Note that much of the prior art focused on the cases with $\omega(t) = 1$, i.e. the cost of a test does not depend on the number of items, and thus the goal is to reduce the number of tests.

Theorem 10. *Suppose there are $m > 1$ items and assume (i)-(iii) holds for the cost function ω. Then for the cost of finding precisely one faulty item with group tests is at least*

$$\Omega \geq \min_{1 \leq x \leq \frac{m}{2}} \omega(x) \left(\log_2 x + \frac{m}{x} - 1 \right) \tag{3.11}$$

Proof. Let us sort the tests by descending size, so that T_1 has the largest number of items while T_b has the least: we assume that

$$t_1 \geq t_2 \geq \cdots \geq t_b$$

where $t_i = |T_i|$ denotes the number of items in test T_i.

Also, we may assume that $t_i \leq \frac{m}{2}$ for every i. Indeed, a test set T_i with $|T_i| \geq \frac{m}{2}$ can be replaced by its complementary set $E \setminus T_i$. The resulting test collection still remains separating if the original one was separating.

We build up the $b \times m$ matrix by adding the rows one-by-one, and in each step we count the number of different columns in the matrix. Let f_i denote the number of different columns when the matrix has i rows, i.e. tests $T_1, \ldots, T_i$ are present, the others are not. For convenience we set $f_0 = 1$. Adding a row the number of different columns cannot decrease, thus $f_{i-1} \leq f_i$ for $i = 1, \ldots, b$. As we have a separating system, all the m columns will be different when the last row is added, giving that $f_b = m$.

When we add T_i, the number of different columns is at most doubled, hence $f_i \leq 2f_{i-1}$, or

$$\log_2(f_i) - \log_2(f_{i-1}) \leq 1 \tag{3.12}$$

for $i = 1, \ldots, b$.

Similarly, by adding test T_i to the collection $T_1, \ldots, T_{i-1}$ can increase the number of different columns in the matrix by at most t_i, giving $f_i \leq f_{i-1} + t_i$, or

$$\frac{f_i - f_{i-1}}{t_i} \leq 1 \tag{3.13}$$

for $i = 1, \ldots, b$.

Now fix an integer k with $1 \leq k < b$. We have

$$\Omega = \sum_{i=1}^{b} \omega(t_i) \geq \sum_{i=1}^{k} \omega(t_i) \left(\log_2(f_i) - \log_2(f_{i-1}) \right) + \sum_{i=k+1}^{b} \omega(t_i) \left(\frac{f_i - f_{i-1}}{t_i} \right). \tag{3.14}$$

We used (3.12) in the first sum, and (3.13) in the second.

The sequence $\omega(t_i)$ is nonincreasing for $i = 1, \ldots, b$ by (i) and our numbering of the tests, hence

$$\sum_{i=1}^{k} \omega(t_i)(\log_2(f_i) - \log_2(f_{i-1})) \geq \sum_{i=1}^{k} \omega(t_k)(\log_2(f_i) - \log_2(f_{i-1})) = \omega(t_k) \log_2(f_k). \tag{3.15}$$

Similarly, the sequence $\frac{\omega(t_i)}{t_i}$ is nondecreasing because of (iii) and our numbering of the tests, giving that

$$\sum_{i=k+1}^{b} \omega(t_i) \left(\frac{f_i - f_{i-1}}{t_i} \right) \geq \sum_{i=k+1}^{b} \omega(t_{k+1}) \left(\frac{f_i - f_{i-1}}{t_{k+1}} \right) = \frac{\omega(t_{k+1})}{t_{k+1}} (m - f_k). \tag{3.16}$$

By substituting (3.15) and (3.16) into (3.14), we have

$$\Omega \geq \omega(t_k)\log_2(f_k) + \frac{\omega(t_{k+1})}{t_{k+1}}(m - f_k) \geq \omega(t_k)\left(\log_2(f_k) + \frac{m - f_k}{t_k}\right). \tag{3.17}$$

This inequality is valid for any k with $1 \leq k < b$. Let we set now k to be the first index j for which $t_j \leq f_j$. Such index clearly exists and $k < b$ because $f_{b-1} \geq \frac{m}{2}$, while $t_i \leq \frac{m}{2}$ for every i. We need to consider two cases:
(1) If $f_{k-1} \leq t_k$, then we have

$$\Omega \geq \omega(t_k)\left(\log_2(f_k) + \frac{m - f_k}{t_k}\right).$$

Note that $t_k \leq f_k \leq 2f_{k-1} \leq 2t_k$, hence for δ defined by $f_k = t_k + \delta$ we have $0 \leq \delta \leq t_k$. Moreover,

$$\Omega \geq \omega(t_k)\left(\log_2(t_k + \delta) + \frac{m - t_k - \delta}{t_k}\right) = \omega(t_k)\left(\log_2(t_k + \delta) - \frac{\delta}{t_k} + \frac{m}{t_k} - 1\right). \tag{3.18}$$

On the interval $0 \leq x \leq 1$ we have the inequality $x \leq \log_2(1 + x)$. We apply this for $x = \frac{\delta}{t_k}$. Note that $0 \leq \delta \leq t_k$ implies that $0 \leq x \leq 1$. We obtain the following inequality

$$\log_2(t_k + \delta) - \frac{\delta}{t_k} \geq \log_2(t_k + \delta) - \log_2\left(1 + \frac{\delta}{t_k}\right) = \\ = \log_2\left(\frac{t_k + \delta}{1 + \frac{\delta}{t_k}}\right) = \log_2\left(\frac{t_k + \delta}{\frac{t_k + \delta}{t_k}}\right) = \log_2(t_k). \tag{3.19}$$

Substituting (3.19) into (3.18) we get

$$\Omega \geq \omega(t_k)\left(\log_2(t_k) + \frac{m}{t_k} - 1\right).$$

(2) If $t_k < f_{k-1}$, then $k > 1$ because $f_0 = 1$ by definition. Thus $f_{k-1} < t_{k-1}$ and based on (3.17) we have

$$\Omega \geq \omega(t_{k-1})\log_2(f_{k-1}) + \frac{\omega(t_k)}{t_k}(m - f_{k-1}).$$

Since $\frac{\omega(t)}{t}$ is a nonincreasing function of t, we have

$$\Omega \geq \omega(f_{k-1})\log_2(f_{k-1}) + \frac{\omega(f_{k-1})}{f_{k-1}}(m - f_{k-1}) \geq \omega(f_{k-1})\left(\log_2(f_{k-1}) + \frac{m}{f_{k-1}} - 1\right). \tag{3.20}$$

In both cases there is an integer x in the interval $[1, \frac{m}{2}]$ such that

$$\Omega \geq \omega(x)\left(\log_2(x) + \frac{m}{x} - 1\right).$$

This is because $f_{k-1} < t_{k-1} \leq \frac{m}{2}$ and $t_k \leq \frac{m}{2}$ hold. This proves the theorem. □

3.3.3 Improved Lower Bound for Sparse Graphs

First let us prove two lemmas needed for the next theorem.

Lemma 9. *Let $m \geq 2$ fixed positive real number. Then $f(t) = \frac{2t}{t+1}\left(\log_2(t) + \frac{m-t}{t}\right)$ is a decreasing function of t for $1 \leq t \leq \frac{m}{2}$.*

Proof. One can verify the lemma directly for $m < 16$. For $m \geq 16$ we have

$$f'(t) = \frac{2}{(t+1)^2}\left(\log_2 t + \frac{m-t}{t}\right) + \frac{2t}{t+1}\left(\frac{1}{t\ln 2} - \frac{m}{t^2}\right),$$

and

$$\frac{(t+1)^2}{2}f'(t) = \log_2 t + \frac{m-t}{t} + \frac{t+1}{\ln 2} - \frac{m(t+1)}{t} = \log_2 t + \frac{t+1}{\ln 2} - m - 1. \tag{3.21}$$

We have to show that

$$\log_2 t + \frac{t+1}{\ln 2} - m - 1 \leq 0 \tag{3.22}$$

on $[1, \frac{m}{2}]$. As the function of t on the left hand side is increasing on the interval, it is enough to verify (3.22) for $t = m/2$. This follows by noting that the function $h(m) := \log_2 m - 2 + \frac{m/2+1}{\ln 2} - m$ is decreasing for $m > 8$, and $h(16) < 0$. □

Lemma 10. *Let $n, m > 0$ be fixed real numbers. Then $\frac{2t}{n+1}\left(\log_2 t + \frac{m-t}{t}\right)$ is an increasing function of t for on $[1, \infty]$.*

Proof. Clearly it is enough to show that $g(t) = t\left(\log_2 t + \frac{m-t}{t}\right)$ is increasing. We have

$$\frac{d}{dt}g(t) = \log_2 t + \frac{m-t}{t} + t\left(\frac{1}{t\ln 2} - \frac{m}{t^2}\right) =$$

$$= \log_2 t + \frac{1}{\ln 2} - 1 > \log_2 t + 0.44 > 0$$

on $[1, \infty]$. □

Theorem 11. *The total cover length for an NL-UFL solution is at least*

$$\|\mathcal{T}\| \geq \begin{cases} \frac{|V|\cdot|E|}{|E|+2}\log_2|E|, & \text{for } |V|-1 \geq \frac{|E|}{2}, \quad (3.23a) \\ |E| + (|V|-1)\log_2\left(\frac{|V|-1}{2}\right) & \text{otherwise.} \quad (3.23b) \end{cases}$$

Proof. Let $\omega(|T_i|)$ be a cost function for m-trail T_i as follows

$$\omega(|T_i|) = \begin{cases} \frac{2|T_i|}{1+|T_i|} & \text{if } |T_i| \leq |V|-1 \quad , \quad (3.24a) \\ \frac{2|T_i|}{|V|} & \text{otherwise.} \quad (3.24b) \end{cases}$$

Next we take (3.5) to get

$$\|\mathcal{T}\| = \sum_{i=1}^{b}\sum_{v=1}^{|V|}\omega_{v,i} = \sum_{v=1}^{|V|}\left(\sum_{i=1}^{b}\omega_{v,i}\right) = \sum_{v=1}^{|V|}\left(\sum_{i|v\in T_i}\omega_{v,i}\right) \geq \sum_{v=1}^{|V|}\left(\sum_{T_i|v\in T_i}\frac{\omega(|T_i|)}{2}\right) \geq \frac{|V|\Omega}{2} \tag{3.25}$$

where Ω is a lower bound on $\sum_i^b \omega(|T_i|)$. The first inequality is a consequence of (3.2). Note that the function $\omega(t)$ satisfies the conditions in the Definition 1, because $\omega(t+1) \geq \omega(t)$, $\omega(1) = 1$, and $\frac{\omega(t+1)}{t+1} \leq \frac{\omega(t)}{t}$. Although the problem scenario with m-trails is slightly different than finding a separating system as in the m-trail problem since none of the items can have all zero code. However, such a constraint further restricts the problem, thus the lower bounds derived in Theorem 10 remains valid here as well.

Consider the case (3.23a). Then we have $t = |T_i| \leq \frac{|E|}{2} \leq |V| - 1$, hence the cost function here is (3.24b). By Theorem 10 we have

$$\Omega \geq \min_{1 \leq t \leq \frac{|E|}{2}} \frac{2t}{1+t} \left(\log_2 t + \frac{|E|}{t} - 1 \right) \tag{3.26}$$

where inside the min there is an decreasing function of t as proved in Lemma 9. Thus, it leads to

$$\Omega \geq \frac{2\frac{|}{E}|2}{\frac{|}{E}|2+1} \left(\log_2 \left(\frac{|}{E}|2 \right) + \frac{|E|}{\frac{|E|}{2}} - 1 \right) = \frac{2|E|}{|E|+2} \left(\log_2 |E| - 1 + 2 - 1 \right) = \frac{2|E|}{|E|+2} \log_2 |E|. \tag{3.27}$$

Putting it together with (3.25) we get (3.23a).

We prove (3.23b) by applying Theorem 10. We obtain

$$\Omega \geq \min_{1 \leq t \leq \frac{m}{2}} \omega(t) \left(\log_2 t + \frac{m}{t} - 1 \right) = \min \left\{ \min_{1 \leq t \leq n-1} \frac{2t}{1+t} \left(\log_2 t + \frac{m}{t} - 1 \right), \min_{n-1 \leq t \leq \frac{m}{2}} \frac{2t}{n} \left(\log_2 t + \frac{m}{t} - 1 \right) \right\} \tag{3.28}$$

where inside the first min there is a decreasing function of t according to Lemma 9, while inside the second min there is an increasing function of t according to Lemma 10. The minimum is attained at $t = n - 1$. It leads to

$$\Omega \geq \frac{2(n-1)}{n} \left(\log_2 (n-1) + \frac{m}{n-1} - 1 \right) = \frac{2}{n} \left((n-1) \log_2 (\frac{n-1}{2}) + m \right) \tag{3.29}$$

Putting it together with (3.25) we get (3.23b). □

Corollary 2. *Let T be a valid m-trail solution for NL-UFL. We have*

$$\|\mathcal{T}\| \geq \frac{|V|}{2} \log_2 (|E| + 1).$$

3.3.4 Lower Bound for Dense Graphs

Theorem 12 provides a lower bound on $\|\mathcal{T}\|$ as a linear function of $|E|$. With a single link failure to be identified, any link must be traversed by at least one m-trail. A link is called *singular* if it is traversed by exactly one m-trail. The following two lemmas are related to singular links.

Lemma 11. *An m-trail T_j of a valid m-trail solution could traverse no more than one singular link.*

Proof. Let links e and e' be two singular links on T_j in a valid m-trail solution. Since T_j is the only m-trail traversing the two links, the two links cannot be distinguished when failure occurs to either one of the links. Thus the m-trail solution is not valid, which contradicts the assumption. □

Lemma 12. *If T_j traverses a singular link, then T_j must be a spanning subgraph (i.e., connected to all the nodes) with $|T_j| \geq |V| - 1$.*

Proof. Let T_j be an m-trail of a valid m-trail solution. If e is a singular link of T_j, and v is a node not in T_j, then the failure of e cannot be detected by v via any m-trail. This contradicts the fact that the m-trail solution is valid. Since T_j is a spanning subgraph, we have $|T_j| \geq |V| - 1$. □

Theorem 12. *Let $\mathcal{T}$ be a valid m-trail solution for NL-UFL on a connected graph with $|E|$ links. We have*

$$\|\mathcal{T}\| \geq 2|E| \left(1 - \frac{1}{|V|}\right)$$

Proof. Let the number of singular links be denoted as σ. Clearly, there is at least a number of σ m-trails according to Lemma 11. The cover length of an m-trail solution can be estimated in terms of σ as follows:

$$\|\mathcal{T}\| \geq 2(|E| - \sigma) + \sigma = 2|E| - \sigma. \tag{3.30}$$

A direct consequence of the Lemma 11 and 12 is:

$$\|\mathcal{T}\| \geq \sigma(|V| - 1). \tag{3.31}$$

In case $\sigma \geq \frac{2|E|}{|V|}$, according to Eq. (3.31) we have

$$\|\mathcal{T}\| \geq (|V| - 1) \cdot \frac{2|E|}{|V|} = 2|E| \left(1 - \frac{1}{|V|}\right).$$

While in case $\sigma < \frac{2|E|}{|V|}$, according to Eq. (3.30) we have

$$\|\mathcal{T}\| \geq 2|E| - \sigma \geq 2|E| \left(1 - \frac{1}{|V|}\right).$$

Thus, for any value of σ the statement holds. □

3.3.5 Line Graphs

The line graph P_n has nodes $v_1, \ldots, v_n$, and the links are (v_i, v_{i+1}) for $i = 1, \ldots, n-1$.

Theorem 13. *The optimal cover length for line graph P_n with n nodes and $|E| = n - 1$ links is $|E|^2$.*

Proof. For P_n the m-trails $T_{1,2}, \ldots, T_{1,n}; T_{2,n}, \ldots T_{n-1,n}$ form a valid m-trail solution for NL-UFL, where $T_{i,j}$ is the subpath form v_i to v_j.

If (v_i, v_{i+1}) is the faulty link, then at v_1 we recognize this correctly from the fact that $T_{1,i}$ is on, while the other $T_{1,i+1}$ is off. Every node between v_1 and v_i can tap these m-trails

and thus achieve L-UFL. $T_{i+1,n}$ and $T_{i,n}$ can be used similarly at the right side of the line graph, in particular at v_n.

The above m-trail solution is optimal in the sense that if a feasible solution does not contain $T_{1,i}$ for some i (or an $T_{j,n}$ for some j) then we cannot have NL-UFL for P_n. If $T_{1,i}$ is not in the m-trail solution, then at v_1 we cannot distinguish the failure of (v_{i-1}, v_i) from the failure of (v_i, v_{i+1}), if $i < n$. If $T_{1,n}$ is not in the m-trail solution then v_1 cannot tell the difference between the errorless state of P_n and the failure of (v_{n-1}, v_n). Same considerations apply to all the paths $T_{j,n}$.

The cost of the collection $T_{1,2}, \ldots, T_{1,n}; T_{2,n}, \ldots T_{n-1,n}$ is

$$1+2+\cdots+n-1+1+2+\cdots n-2 = \frac{n(n-1)}{2}+\frac{(n-1)(n-2)}{2} = (n-1)^2 = |E|^2. \quad (3.32)$$

□

3.3.6 Stars

Let the nodes and links of star S_n be denoted by $v_c, v_1, \ldots, v_{n-1}$, and (v_c, v_i) for $i = 1, \ldots, n-1$, respectively. We note the simple facts that every set of links of S_n spans a connected subgraph; every link is adjacent to the center v_c; and a link set is adjacent to v_i, also referred to as a leaf node, iff it contains the link (v_c, v_i). These observations imply that any set of links is a valid m-trail, and thus the problem is simplified to a coding process for the links without considering whether a set of links can form an m-trail. We will provide a lower and an upper bound on the cover length, which are exactly the same in case $|E| = 2^{b'}$ where b' is an integer.

Lemma 13. *A feasible m-trail solution on S_n has a lower bound on the total cover lengths: $\|\mathcal{T}\| \geq |E|\lceil \log_2(|E|+1)\rceil$.*

Proof. For any node in S_n, it needs to be supported by at least $\lceil \log_2(|E|+1)\rceil$ m-trails (each providing a bit of information in its alarm code table) to uniquely identify $|E|+1$ possible states in the network. This is according to the binary coding mechanism to perform UFL in a graph with m links. Since each node v_i is adjacent to a single link for $i = 1, \ldots, n-1$, the number of m-trails traversing through each link is at least $\lceil \log_2(|E|+1)\rceil$; and the total cover length is $\|\mathcal{T}\| \geq |E|\lceil \log_2(|E|+1)\rceil$ since there are m such links. Thus, the lemma is proved. □

In the following an m-trail construction is developed which exactly yields a solution with $\|\mathcal{T}\| = |E|(\lceil \log_2 |E|\rceil + 1)$.

Lemma 14. *A m-trail allocation construction can be found to achieve NL-UFL in S_n with cover length $\|\mathcal{T}\| \leq |E|(\lceil \log_2 |E|\rceil + 1)$.*

Proof. The proposed m-trail construction is introduced as follows. Let the alarm code a_e be a bit vector of length b assigned to link e, $\forall e \in E$ such that the i-th bit is 1 if $e \in T_i$, and 0 if $e \notin T_i$. The construction is to determine how the alarm code for each link should be designed, which can exclusively determine the set of m-trails $\mathcal{T} = \{T_1, T_2, \ldots, T_b\}$.

Let us define $b' = \lceil \log_2 |E|\rceil$. The construction has each alarm code with a length $b = b'+1$. We have the first b' bits uniquely code the $|E|$ links. Such a link coding process must be feasible since $2^{b'} \leq |E|$. The next b' bits will be exactly the complements of the first b' bits allocated to e. For example, if a_e has the first bit as 0, then its $(b'+1)$-th bit

should be 1, and so on. This results in the fact that the m-trail corresponding to the i-th bit position (denoted as T_i) is the complement graph of m-trail $T_{i+b'}$, $\forall 1 \leq i \leq b'$. Finally, the last bit of a_e is 1 for every e. With the $2b'+1$ m-trails, the construction is complete.

To prove the construction yields a feasible solution, our first step is to argue that every node in the star can perform L-UFL. Clearly, the number of 1's in each alarm code is $b'+1$; in other words, exactly $b'+1$ m-trails are terminated at each leaf node. By further considering that each link is uniquely coded in the construction (using the first $b'+1$ bits), we can conclude that each leaf node can obtain sufficient information to perform UFL based on the terminated $b'+1$ m-trails. Similar consideration can be applied to the proof of L-UFL for the center which is traversed by all the $2b'+1$ m-trails.

With the construction, the number of 1's in the alarm codes of all the $|E|$ links is $|E|(b'+1) = |E|(\lceil \log_2 |E| \rceil + 1)$, which stands for the total cover length of the m-trail solution. □

Theorem 14. *The optimal cover length for star graph with n nodes is*

$$\|\mathcal{T}\| = |E|(1 + \lceil \log_2 |E| \rceil)$$

if $|E| = 2^{b'}$, where b' is an integer.

Proof. Based on lemma 13 and lemma 14, we conclude that the construction is optimal when $|E| = 2^{b'}$ because the upper and lower bounds are equal $\|\mathcal{T}\| = |E|(\lceil \log_2(|E|+1) \rceil) = |E|(b'+1)$. □

3.3.7 Complete Graphs

Theorem 15. *The minimal cover length for complete graph K_n with n nodes is*

$$\|\mathcal{T}\| = (|V|-1)^2$$

.

Proof. We prove the theorem with a construction introduced as follows. Let $S(i)$ denote the star in K_n centered at v_i. $S(i)$ simply consists of $n-1$ links adjacent to v_i. Now let us arbitrarily fix a node v_w in K_n. The m-trail solution, $\mathcal{T}$, is composed of a set of stars each centered at $\forall v_i \neq v_w$. Clearly, each star in $\mathcal{T}$ is a connected subgraph in K_n and is taken as an m-trail. It is clear that with the construction, $\mathcal{T}$ has $n-1$ stars each covering $n-1$ links adjacent to the center of the star; thus we have $\|\mathcal{T}\| = (n-1)^2$.

Such $\mathcal{T}$ is a valid m-trail solution for NL-UFL in K_n. Clearly with the construction, every link must be traversed by two m-trails except for the links adjacent to v_w, and each star is a spanning graph in K_n such that each node can access the status of all the m-trails. Thus, the failure on any link, say (v_i, v_j), can be uniquely identified by any node in K_n due to either (1) the off status of $S(v_i)$ and $S(v_j)$ if $v_i \neq v_w$ and $v_j \neq v_w$, or (2) the off status of $S(v_i)$ or $S(v_j)$ in case $v_i = v_w$ or $v_i = v_w$, respectively. The validity of the construction is thus proved.

For optimality, by Theorem 12 we have

$$\|\mathcal{T}\| \geq 2m\left(1 - \frac{1}{n}\right) = n(n-1)\left(1 - \frac{1}{n}\right) = \\ = n(n-1) - (n-1) = (n-1)^2 = (|V|-1)^2. \tag{3.33}$$

Thus we proved the theorem. □

3.3.8 Circulant Graphs

The construction in Section 2.2.6 is a feasible NL-UFL solution because each m-trail spans the whole network following the unique alarm code of each link. Next we show that the proposed construction yields essentially optimal NL-UFL solutions in terms of the total cost.

Corollary 3. *The m-trail construction in Theorem 7 is an essentially optimal NL-UFL solution.*

Proof. The average nodal degree is 4, thus $|E| = 2|V|$.The total monitoring capacity of the construction of Theorem 7 is $|V|\lceil \log_2(2|V|+1)\rceil$. According to Theorem 11 the total cost is at least

$$|E| + (|V|-1)\log_2\left(\frac{|V|-1}{2}\right) = 2(|V|-1) + 2 + (|V|-1)\left(\log_2(2|V|-2) - 2\right) = 2 + (|V|-1)\left(\log_2(2|V|-2)\right) \quad (3.34)$$

Clearly, the gap is at most $2\log_2(|E|)$. □

3.4 The RSTA-GLS Heuristic Approach for NL-UFL

The section presents a novel heuristic algorithm to solve the NL-UFL m-trail allocation problem in general 2-connected graphs. Inspired by the optimal solution for complete graphs in the previous section, we have observed the great suitability of using spanning subgraphs as the m-trails with bi-directional lightpaths. Thus, the proposed approach uses spanning trees as basic structures of m-trails. To be specific, the proposed approach is based on two novel mechanisms referred to as *random spanning tree assignment* (RSTA) and *greedy link swapping* (GLS), aiming to take the best advantage of flexible and cost-effective spanning tree structures, so as to overcome the topology diversity and maximize sharing of information in solving the m-trail allocation problem for NL-UFL.

With RSTA, b randomly generated spanning trees are launched in the network. The set of b randomly generated spanning trees, denoted as $\mathcal{T}' = [T_1, T_2, \ldots, T_b]$, is used to determine the initial assignment of alarm codes for every link e (denoted as a_e) where the alarm code a_e has the j-th bit as 1 if T_j traverses through e, and 0 otherwise. Clearly, a_e is of length b bits. We claim that $\mathcal{T}'$ can be taken as a valid m-trail solution for NL-UFL if $a_e \neq a_f, \forall e \neq f \in E$. Note that the global code uniqueness, as well as the spanning nature of each T_j, sufficiently guarantees the validity of the solution for L-UFL at each node.

Let us define a *collision* of two codes if they are identical and used by at least two links. Let the bitwise pair at the i-th position of a_e be denoted as $a_{e,[i]}$, which is the code with all identical bits as a_e except for the i-th bit. For example, 011100 is the bitwise pair for the third position of 010100.

The GLS aims to remove all possible collisions by swapping collided codes with unused ones while maintaining the spanning nature of each m-trail. The GLS is performed iteratively upon each bit position $i = 1, \ldots, b$ one by one, where two tasks are defined in each iteration. (1) The GLS checks each bit of a link code a_e under collision, and swaps the link code with $a_{e,[j]}$ if it can resolve the code collision. (2) The code swapping in the first task will turn T_i into a subgraph with either a cycle or two isolated components. Thus GLS swaps another link code a_f on the i-th position with $a_{f,[j]}$ which is currently unused such that T_i is retained as a spanning tree.

3.4.1 Algorithm Description

Algorithm 1: M-Trail Design Problem for L-UFL

```
   Input: G = (V, E)
   begin
1  |  Set b_ini := min{⌈log2(|V| − 1)⌉ + 1, ⌈log2(|E| + 1)⌉}
   |  for b := b_ini to |V| − 1 do
2  |  |  RSTA: Randomly generate b spanning trees
3  |  |  Sort the alarm codes in descending order
4  |  |  for i := 1 to 500 do
   |  |  |  for iterate through the sorted alarm codes do
5  |  |  |  |  if link e and f have the same code then
6  |  |  |  |  |  call GLS(G, e, i)
   |  |  |  |  end
   |  |  |  end
   |  |  |  if every link has a unique alarm code then
7  |  |  |  |  return succeed
   |  |  |  end
   |  |  end
   |  end
   end
```

Algorithm 1 shows the pseudo code of the proposed heuristic algorithm. With each m-trail as a spanning subgraph, the cover length is at least $\|\mathcal{T}\| \geq b(|V|-1)$; therefore it is easy to see that a smaller b leads to a smaller total cover length, at the expense of smaller opportunities for successful GLS due to a smaller number of unused codes that can be taken for replacement. Two lower bounds on b are identified: the first is $b \geq \lceil \log_2(|E|+1) \rceil$, which comes from the fact that each link must have a unique nonzero alarm code; the second is $2^{b-1} \geq |V|-1$, which can be argued in the following lemmas.

Lemma 15. $2^{b-1} \geq |T_j| \geq |V|-1$ *holds for* $j = 1, \ldots, b$.

Proof. Since T_j is a spanning sub-graph, we have $|T_j| \geq |V|-1$. Further, $|T_j|$ is upper bounded by the number of codes with 1s at the bit position j, which is nonetheless no more than half of the 2^b according to the binary coding mechanism. Thus we have $2^{b-1} \geq |T_j| \geq |V|-1$. □

With the two lower bounds, the initial value of b in Step (1) is set as

$$b = \min\{\lceil \log_2(|V|-1) \rceil + 1, \lceil \log_2(|E|+1) \rceil\}.$$

In case the algorithm has never succeeded with b, the algorithm starts over again by increasing b, and this iteration continues until either we find a valid solution in Step (7) or $b = |V|$. With such initial setting of b, the minimum achievable cover length $||\mathcal{T}^*||$ can be expressed as:

$$||\mathcal{T}^*|| \geq (|V|-1) \min\{\lceil \log_2(|V|-1) \rceil + 1, \lceil \log_2(|E|+1) \rceil\} \tag{3.35}$$

In Step (2), b random spanning trees are generated, and each link is assigned with a code of length b where the i-th bit is 1 if the link is traversed by T_i. In our implementation the method of Aldous/Broder [66, 67] is adopted for this purpose. The code assignment in Step (2) may cause code collision (i.e., a code assigned to two or multiple links); and some links may not be traversed by any spanning tree (with an all-zero code "$00\ldots0$"). From

Step (3) to (6), the collided codes are identified by sorting all the codes. For each link pairs with a collided code, the GLS method is called in Step (6) to resolve the collision. The algorithm stops if the loop in Step (4) is executed over 500 times to avoid infinite loops.

Algorithm 2: Greedy Link Swapping (GLS)

```
   Input: G = (V, E) link e with collided code and iterator i
   begin link e has code a_e = {a_{e,[1]}, ..., a_{e,[b]}}
6.1 |  for i := 1 to b do
    |    if a_{e,[j]} is currently unused and nonzero then
6.2 |      if a_{e,[i]} = 0 then
    |        After adding e into T_i the m-trail has a cycle. Let L be the path in m-trail T_i
    |        between the adjacent nodes of e
6.3 |
    |      else  a_{e,[i]} = 1
    |        After erasing e in T_i the m-trail falls into two parts. Let L be the set of edges in
    |        G that can be used to reconnect the parts
    |      end
6.4 |      for every link f ∈ L do
    |        if a_{f,[j]} is unused and nonzero then
6.5 |          flip the bits: a_{e,[i]} = ¬a_{e,[i]} and a_{f,[i]} = ¬a_{f,[i]}
    |          return succeed
    |        end
    |      end
    |      if i > 250 ∧ sparse(G) ∧ a_{e,[i]} = 0 then
6.6 |        flip the bits: a_{e,[i]} = ¬a_{e,[i]}; return succeed
    |      end
    |    end
    |  end
    |  return not succeed
   end
```

Algorithm 2 is called when a collision is found at link e, and the goal of the algorithm is to find an unused non-zero alarm code that can replace the old one and resolve the collision. In Step (6.1) we inspect each bit position on a_e to see if there is a nonzero and unused bitwise pair. If there is such $a_{e,[j]}$ (i.e., the bitwise pair of a_e at the i-th position), then we check how such swapping of the code pair would impact T_i. In case the swapping is to flip the bit at the i-th position from 0 to 1 (in Step (6.2)), it means the swapping simply adds link e to T_i. Since T_i is a spanning tree, adding e to T_i must create a loop for T_i. Thus, the algorithm tries to remove the loop by inspecting each link along the loop (denoted as L) except for e, to see if removing any link along L is possible, as shown in Step (6.4). If there is any link $f \in L$ for which $a_{f,[j]}$ is unused and nonzero, the pair of codes are swapped to remove the link from T_i such that T_i is still a spanning tree in Step (6.5).

In case the swapping is to flip the bit at the i-th bit position from 1 to 0 (in Step (6.3)), the swapping will remove e from T_i. Since T_i is a spanning tree where each node pair is at most connected by a single link, removing e from T_i must break T_i into two isolated components. Then, the GLS needs to reconnect the two components. The links that can be used to reconnect the two isolated components except for e (denoted as L), are inspected one after the other, until any link f in L with an unused non-zero $a_{f,[j]}$ is found. If such link f is found successfully, the code pair is swapped in order to retain T_i as a spanning tree in Step (6.5).

In our implementation, in case the problem cannot be solved in 250 iterations (which happens when the networks are very sparse), cycles are also allowed. We call a network

sparse, i.e. $sparse(G) := true$, iff the lower bound by Theorem 9 is sharper than the bound of Theorem 12. As shown in Step (6.6), a_e can be swapped with $a_{e,[j]}$ if a_e is nonzero and unused with 0 at the i-th position.

Note that in case the algorithm fails to find a way to resolve a code collision under a specific code length b, it breaks the loop and adds one more bit in the alarm code. It implies that the number of unused nonzero codes is doubled, which significantly helps the algorithm to resolve any possible code collision. We will show in the simulation that the proposed heuristic returns a valid NL-UFL solution in all the randomly generated topologies (over 2000).

3.4.2 An Illustrative Example

First let us illustrate the algorithm through an example using the graph in Fig. 3.3. Four random spanning trees are generated first (i.e., $b = 4$), each corresponding to an m-trail as shown in Fig. 3.3a. Clearly, these m-trails explicitly define an alarm code for each link as shown on Fig. 3.3a. The four m-trails do not form a valid m-trail solution due to the collisions $a_{(v_3,v_4)} = a_{(v_0,v_1)} = 0010$ and $a_{(v_5,v_3)} = a_{(v_3,v_1)} = 1011$.

The GLS first tries to swap $a_{(v_0,v_1)}$ with its bitwise pair of the first bit position, i.e., 1010, to remove the collision between $a_{(v_3,v_4)}$ and $a_{(v_0,v_1)}$. It is feasible since 1010 is currently unused. But with the swapping T_1 will no longer be a spanning tree due to the cycle $L_1 = (v_0, v_1, v_3, v_2)$. Thus, a link from L_1 should be removed, and the GLS does this by inspecting each link along L_1 except for the link (v_0, v_1). In our case, the first link inspected is (v_0, v_2) with an alarm code 1101. Since $a_{(v_0,v_2),[1]} = 0101$ has already been taken by link (v_5, v_4), we proceed to the next link (v_3, v_2) in L_1. Luckily, the bitwise pair $a_{(v_3,v_2),[1]} = 0111$ is currently unused by any link. Since both of the tasks in the iteration are feasible, the GLS then finalizes the swapping attempts for $a_{(v_0,v_1)}$ and $a_{(v_3,v_2)}$ with their unused bitwise code pairs: $a_{(v_0,v_1),[1]} = 1010$ and $a_{(v_3,v_2),[1]} = 0111$, respectively. As a result, T_1 is reshaped as shown in Fig. 3.3b.

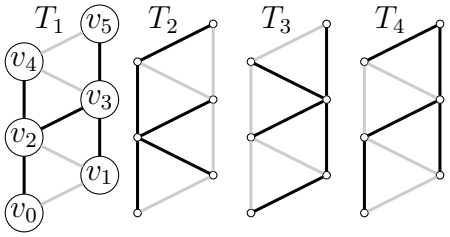
For the collision between $a_{(v_5,v_3)}$ and $a_{(v_3,v_1)}$, the GLS first finds the bitwise pair of $a_{(v_5,v_3)}$ at the first bit position as 0011 that is currently unused. But by swapping $a_{(v_5,v_3)}$ as 0011, T_1 breaks into two components: $\{v_0, v_1, v_2, v_3, v_4\}$ and $\{v_5\}$, and the only way to reconnect them is to swap $a_{(v_5,v_4)} = 0101$ by its bitwise pair at the first bit position: 1101, which, unfortunately, has already been used. Therefore, there is no simple solution at the first bit, and the algorithm turns to find a solution from T_2. The bitwise pair of $a_{(v_5,v_3)} = 1011$ at the second bit position is 1111, which is unused. Further, due to the swapping, a loop $L_2 = (v_2, v_3, v_5, v_4)$ is formed on T_2 and needs to be removed. The GLS inspects each link on L_2 except (v_5, v_3), where the bitwise pair of $a_{(v_5,v_4)} = 0101$ at the second bit position, 0001 is currently unused. Thus, the GLS concludes the feasibility of the swapping for $a_{(v_5,v_3)}$ and $a_{(v_5,v_4)}$ with their unused bitwise pairs $a_{(v_5,v_3),[2]} = 0011$ and $a_{(v_5,v_4),[2]} = 0001$, respectively.

Finally, the GLS completely solved the code collisions, and the final result is shown in Fig. 3.3c, which is a valid m-trail solution for NL-UFL.

3.4.3 Performance Verification of RSTA-GLS

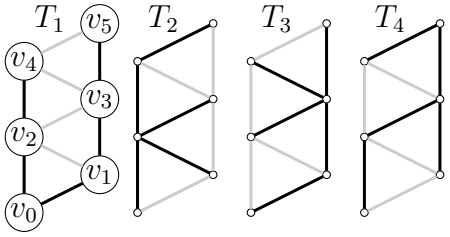
Simulations on thousands of randomly generated planar 2-connected backbone network topologies via LEMON [59] were conducted. With LEMON random graph generator, nodes are firstly allocated into an area of unit square with a uniform distribution, and links with small physical length are added to keep the graph planar if possible, and to

link	code
(v_3, v_4)	0010
(v_0, v_1)	0010
(v_1, v_2)	0100
(v_5, v_4)	0101
(v_5, v_3)	1011
(v_3, v_1)	1011
(v_2, v_4)	1100
(v_0, v_2)	1101
(v_3, v_2)	1111



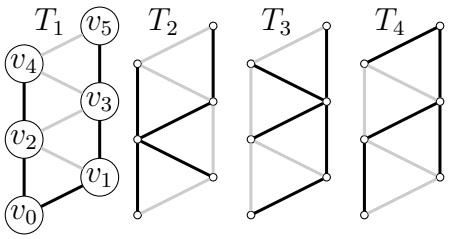
(a) Four random spanning trees

link	code
(v_3, v_4)	0010
(v_1, v_2)	0100
(v_5, v_4)	0101
(v_3, v_2)	0111
(v_0, v_1)	1010
(v_5, v_3)	1011
(v_3, v_1)	1011
(v_2, v_4)	1100
(v_0, v_2)	1101



(b) Swapping $a_{(v_0,v_1)}$ and $a_{(v_3,v_2)}$ with their bitwise pairs at the first position.

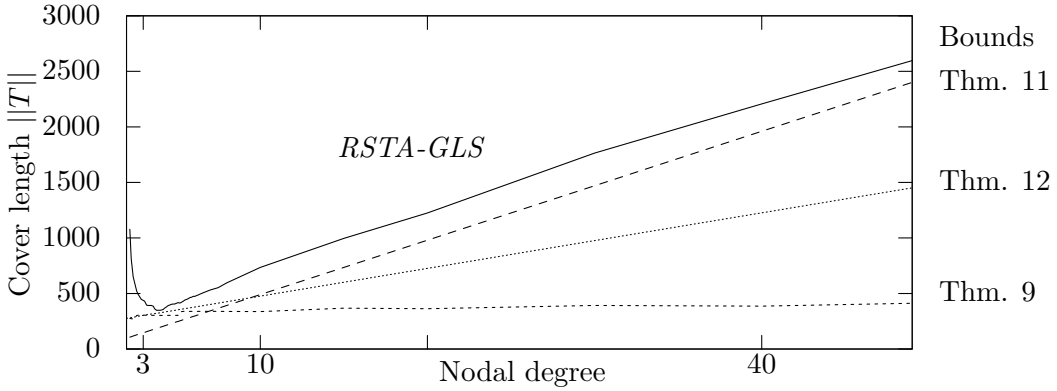
link	code
(v_5, v_4)	0001
(v_3, v_4)	0010
(v_1, v_2)	0100
(v_3, v_2)	0111
(v_0, v_1)	1010
(v_3, v_1)	1011
(v_2, v_4)	1100
(v_0, v_2)	1101
(v_5, v_3)	1111



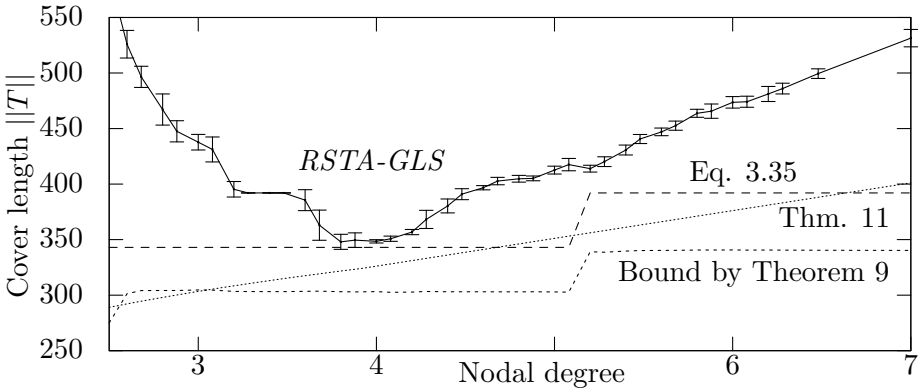
(c) A feasible solution by swapping $a_{(v_5,v_3)}$ and $a_{(v_5,v_4)}$ with their bitwise pairs at the second position.

Figure 3.3: Illustrative example of the proposed algorithm.

keep the facets of the planar graph of equal size (see examples on Fig. 3.5). In such a way thousands of random networks were generated and classified according to their size in number of edges or nodes and nodal degrees. Finally a series of networks are selected and stored respectively, according to their (1) number of nodes, (2) number of links, and (3) nodal degrees. The performance metrics of interest are the total cover length of the solution and the running time. In addition to the evaluation of the proposed heuristic with the derived upper bounds and optimal solutions, the impact of topology diversity to the NL-UFL m-trail problem will be investigated. All three metrics are examined with respect to different network sizes (i.e., the number of nodes or links) and topology densities (i.e., nodal degree), which will be presented in the following three subsections.



(a) Networks with 50 nodes and different number of links.



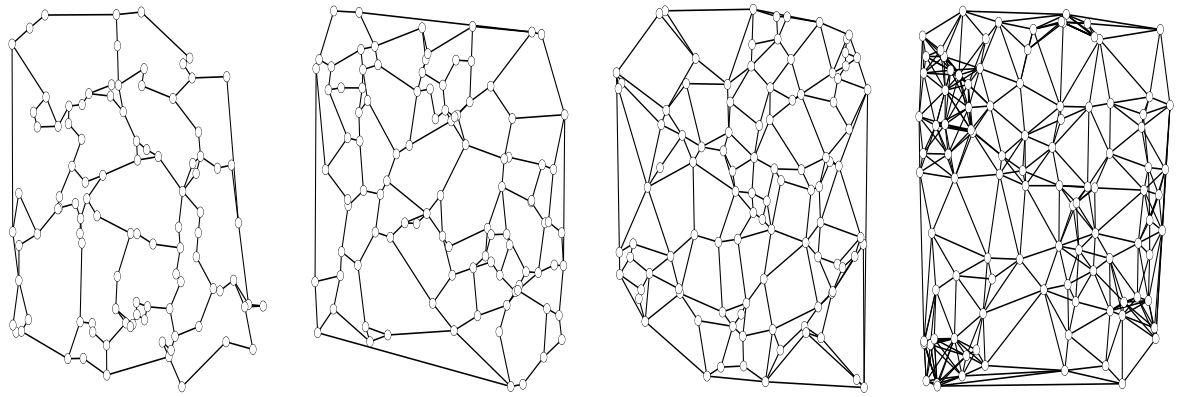
(b) Plot with nodal degree from 2.5 to 7.

Figure 3.4: The cover length versus average nodal degree (900 randomly generated topologies with 50 nodes).

Performance Comparison

We first examine the proposed scheme by comparing it with the derived bounds as shown in Fig. 3.4(a) and Fig. 3.4(b) in term of total cover length (i.e., $\|\mathcal{T}\|$). Fig. 3.4(a) shows the cover length when the average nodal degree of the topologies with 50 nodes is increased from 2.5 to 49, where three curves by the proposed *RSTA-GLS*, Theorem 11, Theorem 9, and Theorem 12, are plotted respectively. It is shown that *RSTA-GLS* yields an average gap of less than 9.7% to the best bound of the two, which is given by Theorem 11 when the topology is sparse while by Theorem 12 in denser ones. Fig. 3.4(b) provides a closer look at the range of nodal degree $[2.5, 7]$ in Fig. 3.4(a), where the bounds given by Theorem 11, Theorem 9 and Eq. (12) are also plotted. Note that Eq. (12) is a bound that intrinsically exists due to the initial assignment of b. To summarize, the proposed *RSTA-GLS* can yield solutions very close to the lower bounds that we derived, and the two bounds in Theorem 11 and Theorem 12 can well account for sparse and dense topologies, respectively. Further, since the bounds derived for general graphs do not consider sufficient topological features, both bounds are not effective when the network nodal degrees are quite low (e.g., less than 3). Nonetheless, the minimal gap is observed when the nodal degree ≈ 4.

Three schemes were implemented and compared in the simulation. *RSTA-GLS* (denoted by $\times$ on the charts) refers to the proposed heuristic. CA_S (denoted by $+$ on the

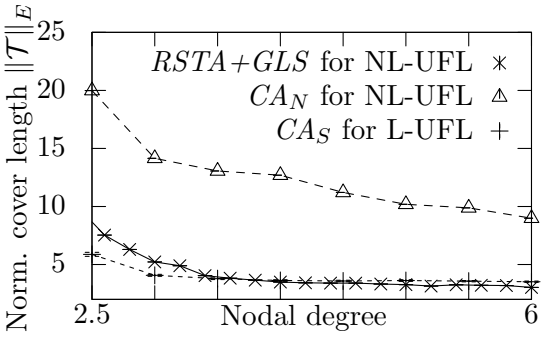


(a) Avg. nodal degree 2.5 (b) Avg. nodal degree 3 (c) Avg. nodal degree 4 (d) Avg. nodal degree 8

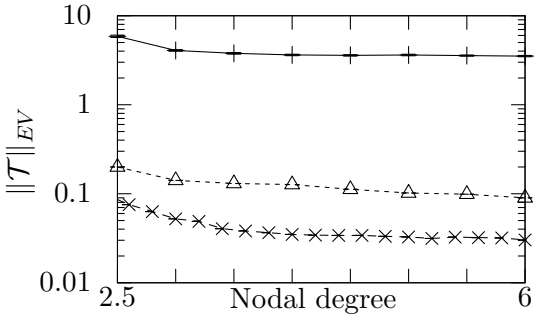
Figure 3.5: Examples of random 100 node backbone networks with different number of links.

Network [68]	Graph			Theorem			$\|\mathcal{T}\|$		$\|\mathcal{T}\|_E$		#m-trails		Time [s]	
	$\|V\|$	$\|E\|$	diam- met	9	11	12	*RSTA +GLS*	CA_N	*RSTA +GLS*	CA_N	*RSTA +GLS*	CA_N	*RSTA +GLS*	CA_N
Pan-Europe	16	22	6	79	65.6	82	107	241	2.43	5.47	7	24	0.39	0.42
German	17	26	6	84	74	97	128	368	2.46	7.07	8	33	0.51	0.65
ARPA	21	25	7	108	91.4	95	140	354	2.80	7.08	6	33	0.36	0.64
European	22	45	5	127	116	171	231	617	2.56	6.85	11	53	2.10	4.59
USA	26	42	8	155	133	161	229	667	2.72	7.94	9	53	2.16	3.99
Nobel EU	28	41	8	168	142	158	248	689	3.02	8.40	7	50	0.87	3.38
Italian	33	56	9	200	184	217	329	1113	2.93	9.93	10	66	4.83	11.1
Cost 266	37	57	8	225	207	221	343	918	3.00	8.05	8	55	2.04	14.4
North Amer.	39	61	10	241	222	237	378	1020	3.09	8.36	8	50	2.31	15.5
NSFNET	79	108	16	579	520	426	760	2634	3.51	12.2	9	100	6.05	227

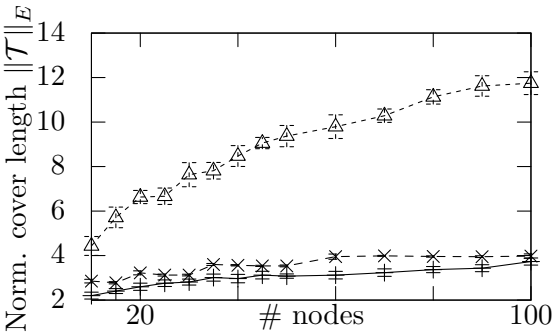
Table 3.1: Results by the proposed *RSTA-GLS* and CA_N on some well-known networks.



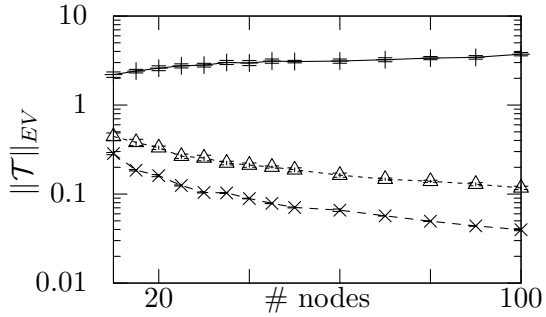
(a) 1560 randomly generated topologies with 100 nodes.



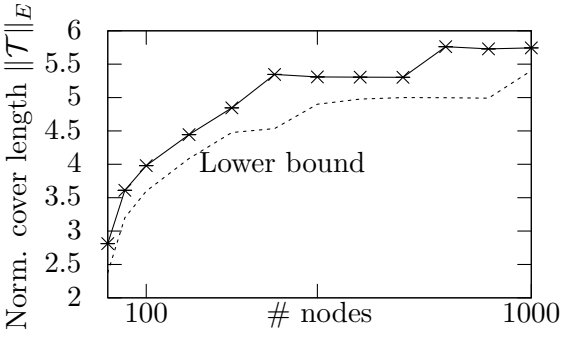
(b) 1560 randomly generated topologies with 100 nodes.



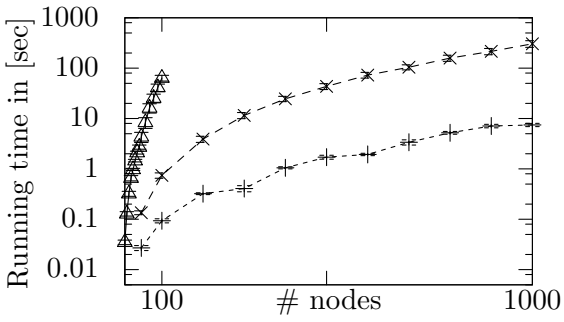
(c) 2400 randomly generated topologies with nodal degree ≈ 4.



(d) 2400 randomly generated topologies with nodal degree ≈ 4.



(e) 240 randomly generated topologies with nodal degree ≈ 4.



(f) The running time of 1640 randomly generated topologies with nodal degree ≈ 4.

Figure 3.6: The normalized cover length and running time by *RSTA-GLS* ($\times$) and CA_S ($+$) and CA_N ($\triangle$).

charts) corresponds to the method that each m-trail is allocated one after the other to distinguish each pair of links using any Dijkstra's algorithm based scheme, such that L-UFL can be achieved at an arbitrarily chosen monitoring node (MN). Such a method is generic and has been considered in a number of previously reported studies [35, 60, 26]. Rather than just for L-UFL at a single MN with CA_S, CA_N (denoted by $\triangle$ on the charts) achieves NL-UFL by performing CA_S sequentially at each node. In our implementation loopback switching is allowed, thus CA_S and CA_N algorithms are modified such that an L-UFL solution for any node can always be found in a connected graph according to Theorem 8.

The comparison results are shown in Fig. 3.6. We normalize the cover length over the number of links and both the numbers of links and nodes, denoted as $\|\mathcal{T}\|_E$ and $\|\mathcal{T}\|_{EV}$, respectively. $\|\mathcal{T}\|_E$ is a measure on the average number of WLs per edge while $\|\mathcal{T}\|_{EV}$ is on the average number of WLs per edge per MN, respectively. Clearly we have $\|\mathcal{T}\|_E = \frac{\|\mathcal{T}\|}{|E|}$ and $\|\mathcal{T}\|_{EV} = \frac{\|\mathcal{T}\|}{|V|\cdot|E|}$.

In Figs. 3.6(a) and (b), we observed that the proposed *RSTA-GLS* consumes much smaller $\|\mathcal{T}\|_E$ and $\|\mathcal{T}\|_{EV}$, respectively, than that by CA_N when the number of network nodes is 100 and the average nodal degree is increased from 2.5 to 6. It is clearly demonstrated that *RSTA-GLS*, which enables all the 100 nodes to individually serve as L-UFL MNs, can achieve similar cover lengths to that by CA_S where only a single L-UFL MN is supported. This demonstrates the effect of on-off status sharing among the on-trail nodes of a common m-trail, such that the increase of the number of MNs would lead to little increase in the total cover length. They also show that $\|\mathcal{T}\|_E$ and $\|\mathcal{T}\|_{EV}$ decrease when the nodal degree is increased (or as the number of links is increased) due to the better connectivity which yields larger design space for allocating the m-trails. Fig. 3.6(c) and (d) plot $\|\mathcal{T}\|_E$ and $\|\mathcal{T}\|_{EV}$ against the number of nodes given the nodal degree ≈ 4, which yield similar comparison results as in Figs. 3.6(a) and (b). Fig. 3.6(e) plots the solutions by *RSTA-GLS* and the best lower bound derived in Section 3.3.1. One can verify that the normalized cover length has a logarithmic relation with the increasing number of nodes (from 100 to 1,000) when the average nodal degree is kept as a constant (i.e., ≈ 4).

Finally, the computation efficiency of the proposed *RSTA-GLS* is examined, which is shown in Fig. 3.6(f). We can clearly see that our scheme yields significantly better computation efficiency than CA_N. Note that CA_S takes shorter computation time since only a single MN is considered, compared with the case where *RSTA-GLS* enables all the nodes as MNs each being able to perform L-UFL. With *RSTA-GLS*, the NL-UFL m-trail problem on a 1000-node network can be solved within 3 minutes.

Table 3.1 provides results of *RSTA-GLS* and CA_N on some well-known network topologies taken from [68]. The number of nodes, links and the diameter in hops of every topology graph is also shown in the first three columns of the table. The results are similar to that on randomly generated graphs as in Fig. 3.4 regarding the number of required m-trails (i.e., $\|\mathcal{T}\|$), average WLs per link (i.e., $\|\mathcal{T}\|_E$), and running time (in seconds) under various network sizes and topology densities.

To summarize the comparison results above, the proposed *RSTA-GLS* is witnessed to achieve desired computation efficiency in handling large networks, and its feasibility in the operation of future all-optical backbone is proved in terms of resource consumption for achieving NL-UFL using bi-directional m-trails. For example, an optical network with 288 links and 100 nodes takes averagely 3.4 WLs along each link, and each MN only consumes approximately 0.034 WLs per link for achieving NL-UFL. Thus, in the case

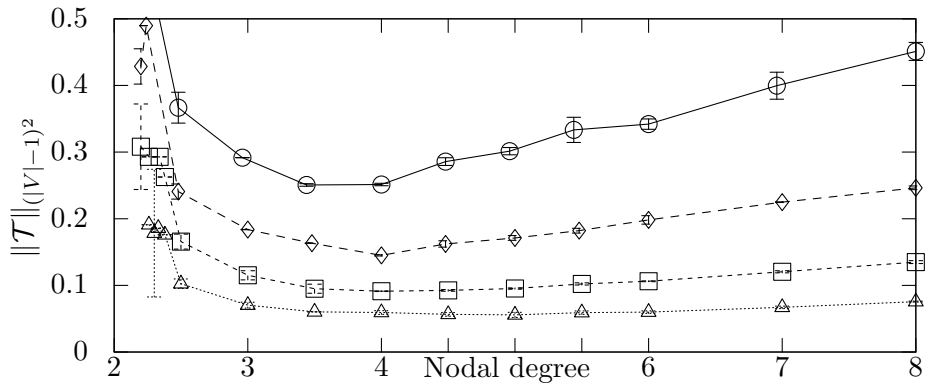


Figure 3.7: Plot of $\|\mathcal{T}\|_{(|V|-1)^2}$ with $n = 25(\circ), 50(\diamond), 100(\square), 200(\triangle)$ (1280 randomly generated topologies.

that each fiber has 100 wavelength channels and 10 fibers bundled in a single conduit, the system needs to spend only 0.68% of the total capacity for achieving NL-UFL of any single conduit.

The Impact of Topology Diversity

Fig. 3.7 shows the performance of the proposed *RSTA-GLS* on topologies with $|V| = 25, 50, 100, 200$ as the increase of the average nodal degree of the topologies. Instead of plotting $\|\mathcal{T}\|$ for each case, we normalize the results on topologies of n nodes by $(|V|-1)^2$ (which is denoted as $\|\mathcal{T}\|_{(|V|-1)^2}$). Note that a line graph P_n and a complete graph K_n represent the least and most densely meshed topology with n nodes, respectively. Both topologies have an optimal solution $\|\mathcal{T}\| = (|V|-1)^2$ as derived in Section 3.3.7, which is supposed to be the largest possible $\|\mathcal{T}\|$ for any topology with n nodes. We expect that all other topologies should yield solutions less than $(|V|-1)^2$. This is attested in the figure, where the curves of $\|\mathcal{T}\|_{(|V|-1)^2}$ is observed as a "V" shape for each specific n value. It also shows that $\|\mathcal{T}\|_{(|V|-1)^2}$ is minimal when the average nodal degree of the topologies is in the range of $[3.5, 4.5]$ of all the values of $|V|$ investigated. This demonstrates that the proposed scheme can be the most suitable to work in optical networks with moderate connectivity.

Chapter 4

An All-Optical Restoration Framework with M-Trails

4.1 Introduction

A general approach to increase availability of each connection is to pre-plan one or multiple *protection lightpaths* (P-LPs) for each *working lightpath* (W-LP). Fortunately, network failures are rare events thus it is a widely accepted strategy to share the allocated spare capacity among multiple P-LP(s) that are assumed not to be activated at the same time. This is also referred to as *shared protection*. In contrast to *dedicated protection*, a shared protection scheme relies on a suite of real-time mechanisms to restore the failed W-LPs, including failure localization, failure notification, failure correlation, and device configuration; and these real-time mechanisms for traffic restoration are also known as *fault management* defined in Generalized Multi-Protocol Label Switching (GMPLS) based recovery [69].

Numerous research efforts have been addressed to the design of protection schemes. Their different flavors include *shared backup path protection* (SBPP), *segment shared protection* (SSP), link protection, failure dependent protection (FDP, or referred to as *path restoration*), and pre-configured protection [70, 71, 63, 72, 73, 74]. Two objectives are widely considered in the design of a protection scheme in optical networks, namely *capacity efficiency* and *fault management complexity*. The former concerns the amount of consumed *spare capacity*, which is the capacity in terms of wavelength channels (WLs) reserved but not necessarily configured for the P-LP(s); while the latter is measured as *restoration time*, which equals to the duration from the instant that the traffic is unexpectedly interrupted to that the traffic is completely restored.

In the effort of modeling the restoration time of an interrupted W-LP, one needs to drill into the GMPLS fault management signaling protocols. In general, the failure is firstly localized at one or multiple nodes adjacent to the failure, and the node(s) send notifications to the corresponding switching node of each affected W-LP, which concludes the failure event by correlating the failure notifications, and restore the working traffic via the corresponding P-LP. When multiple P-LPs are prepared for a W-LP such as in the case of FDP and SSP, a P-LP specific to the failure event should be chosen, and the switching node of the P-LP is notified. Then, the switching node has to initiate a P-LP setup process, mostly via an optical layer resource reservation protocol such as Resource Reservation Protocol (RSVP) extended for GMPLS. The traffic switchover starts once

all the intermediate nodes of the P-LP are well configured, then the failure restoration is completed. Note that under shared protection, the WLs along the P-LP(s) are reserved but may not be pre-configured. It is clear that all the abovementioned tasks should be performed in a sequence, which add up to the total restoration time.

In the spectrum of different flavors of shared protection schemes, there exists a compromise between capacity efficiency and restoration time, in which FDP and pre-configured protection (such as p-Cycle) are the two extremes. With FDP, each connection is assigned with multiple P-LPs, and one is activated for the restoration purpose according to the identified failure event. With stub release, FDP has long been recognized as having the optimal capacity efficiency and widely taken as the performance benchmark in the design of shared protection schemes. However, the FDP restoration process is subject to the highest control and signaling complexity that possibly yields the longest restoration time, mostly because the switching node has to precisely localize the failure event for real-time selection of a P-LP.

The other extreme is p-Cycle. Thanks to pre-configured intermediate nodes along P-LPs, the p-Cycle based schemes achieve very fast restoration due to the fact that the only after-failure action is the reconfiguration of the two nodes responsible for switching and merging the affected working traffic. The switching and merging nodes identify and correlate the failure via in-band monitoring of the W-LP that is assumed to be bi-directional, without waiting for any further failure notification. Thus in contrast to any other flavor of shared protection, the p-Cycle based schemes can minimize the fault management complexity (and the restoration time), and possibly be implemented completely in the optical domain independent from any upper layer control protocol. Such simplicity and fast restoration speed are nonetheless at the expense of consuming the highest redundancy.

It was open whether there is an efficient approach that can facilitate a general protection scheme to achieve an all-optical and signaling-free restoration process like p-Cycle without sacrificing the capacity efficiency of FDP and SSP. In this chapter we intend to investigate a novel framework of fast restoration in all-optical networks, aiming to enable any shared protection scheme to achieve true all-optical restoration without any aid from the upper layer protocol. The proposed new framework incorporates the conventional restoration process with state-of-the-art failure localization techniques, namely *Network wide Local Unambiguous Failure Localization* (NL-UFL) described in Section 3.2.4, which takes advantage of a set of intelligently deployed supervisory lightpaths, called *monitoring trails* (m-trails). The on-off status of the m-trails can be converted into network-wide failure status locally available to each node, such that each node can autonomously respond to any pre-defined failure event and collaboratively work for the recovery of the interrupted W-LPs. The chapter will detail how the restoration process can be realized, and will compare the proposed approach with a couple of p-Cycle based schemes via a case study in terms of restoration time, computation complexity, and capacity efficiency.

4.2 Restoration Time Analysis

Restoration time is an important performance measure of a protection/restoration scheme, which concerns the data loss and service discontinuity in the occurrence of a failure event. The best situation is that an optical domain failure can be completely restored in the optical domain while leaving the upper layer protocols (e.g., Open Shortest Path First (OSPF) and Transport Control Protocol (TCP)) unaware of, or at least with minimum impact due to, the event. To achieve this goal, a restoration process with a few

tens of milliseconds of restoration time is desired.

Here we provide an analysis on restoration processes of generic protection schemes according to the context of GMPLS fault management [69]. GMPLS defines the fault management phase in the restoration of a failed W-LP as composed of a number of real-time tasks, namely failure localization, failure notification, failure correlation, path selection, and device configuration. Formally we have:

$$t_r = t_l + t_n + t_c + t_p + t_d \ , \tag{4.1}$$

Each term is explained as follows. t_l is for failure localization, which is defined as the time between the instant that the failure is detected at the nodes nearby, and the instant that the nodes send alarms to the corresponding switching node of the W-LP. Note that a failure could cause multiple nodes to alarm.

t_n is for failure notification, which is defined as the period between the instant that the failure event is localized at some nodes, and the instant that the switching node receives the notification. t_n could be significantly reduced if the failure localization is performed at a node physically close to the switching node. For example, a link/span restoration scheme localizes a failure exactly at the switching node, which should yield a negligible t_n. On the other hand, a path protection scheme usually takes much larger t_n due to multi-hop signaling, usually supported by a control plane protocol.

t_c is the time between the instant that the switching node of the W-LP receives the failure notification and that it completes the failure correlation. t_c is non-trivial when multiple notifications are received at the switching nodes due to failure propagation, where the switching node has to spend some time to wait for all possible notifications for a precise failure identification.

t_p is the time for path selection at the switching node when multiple P-LPs are pre-planned (e.g., FDP). It is considered negligible provided we have a precise identification of the failure event.

t_d is defined as the time for setting up the P-LP. The typical time to configure switching matrix of an OXC is 20ms. By using the GMPLS path setup message, device configuration is performed sequentially at every node along the P-LP, thus leading to a latency of tens to hundreds of milliseconds depending on the length and hop counts of the P-LP.

Note that there are certainly other components contributing to the total restoration time such as that for failure detection and traffic resumption, which are nonetheless not considered in our model since they are common to all the protection schemes and indifferent to the proposed framework.

An example is given in Fig. 4.1 for a general restoration process. The W-LP goes through $s - a - b - c - d$, and a P-LP goes along $a - e - f - c$. When a failure hits $b - c$, node b and c localize the failure by taking a period t_l, and send notifications to the corresponding switching node, i.e., node a, by taking a period t_n. Note that the notification may be sent to s instead of a for decision. No matter which node serves as the switching node, it needs to wait for t_c to make sure only b and c are alarming. Then the switching node initiates a resource reservation process for device configuration at nodes a, e, c and f. After being acknowledged, the working traffic is switched at node a and merges back to the W-LP at node c to complete the restoration.

Based on the model, p-Cycle yields about 40ms of restoration time by having $t_l \approx 10$ms for the failure to be localized at the nodes adjacent to the failure, t_c as 10ms mainly for failure correlation, and $t_d = 20$ms required at the two nodes for switch fabric configuration.

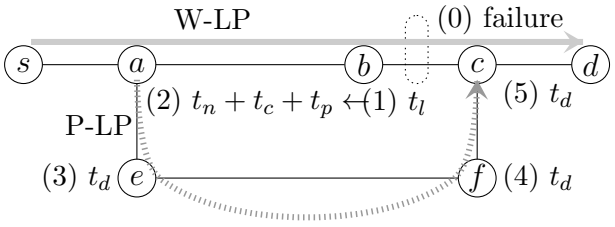


Figure 4.1: An example of conventional restoration process.

Other protection schemes have to significantly rely on a suite of multi-hop signaling mechanisms in their restoration processes. In shared link/span protection, t_l is similar to that of p-Cycle, but establishing a P-LP requires an additional few tens of milliseconds for device configuration. For path protection/restoration schemes such as SBPP, the restoration time can be up to a hundred milliseconds due to the multi-hop signaling notification process and longer P-LPs. Further, the restoration process for FDP yields the longest restoration time due to the following three reasons. (1) Since precise failure correlation is needed at the switching node, $t_n + t_c$ is the longest possible. (2) Multiple P-LPs are in place for each W-LP, yielding non-zero t_p. (3) The P-LPs are not pre-configured, thus a regular P-LP setup process is required.

4.3 Signaling-Free Restoration Framework

The proposed framework is characterized by an integration of the conventional restoration process with a state-of-the-art all-optical failure localization technique, NL-UFL. In a nutshell, NL-UFL creates an all-optical fault management system by taking all the nodes as independent MNs. Without relying on any multi-hop signaling protocol, each node can obtain the on-off status of the traversing m-trail(s) by tapping the optical signals along the m-trail(s), which further facilitate the formation of a valid alarm code that uniquely maps to the failure event. Thus, NL-UFL leads to a truly all-optical and signaling-free failure localization system and is taken as a key functional block in the proposed framework.

Recall the restoration time analysis for the conventional GMPLS based recovery where the restoration time is mainly due to failure localization (t_l), failure notification (t_n), failure correlation (t_c), and device configuration (t_d). On the other hand under the proposed framework, we show that the four terms can be significantly reduced or removed, and the resultant restoration process can be implemented completely in the optical domain without the aid of any network layer control protocol.

Firstly, since the switching node of an interrupted W-LP is aware of the failed SRLG via the on-off status of the traversing m-trails, the time for failure localization is simply the propagation delay of the m-trails interrupted by the failure, while the time for failure notification is completely removed. Since an optical flow is deterministic in terms of its propagation speed, the time for failure correlation can be the minimum. Further, since all the intermediate nodes of the P-LPs corresponding to the failure event can localize the failure event thanks to NL-UFL, they can start configuring their switch fabrics based on the collected alarm code without waiting for P-LP setup request from any other network entity. Thus the W-LP setup latency can be minimized as well. This leads to a completely all-optical and deterministic restoration process.

To be specific, the fault management latency under the proposed restoration process

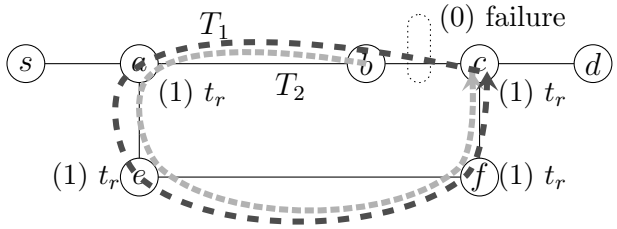


Figure 4.2: The proposed restoration process.

can be modeled as:

$$t_r = t_{nc} + t_d \ ,$$

where t_{nc} and t_d is the time for failure notification/correlation, and the time for device configuration for the P-LP setup, respectively. Different from that in the conventional restoration process, t_{nc} is determined solely by the light propagation delay of the m-trails. On the other hand, t_d should be close to the time for OXC configuration at a single node, since all the intermediate nodes of the P-LP can configure their OXCs almost in parallel.

Following up Fig. 4.1, Fig. 4.2 is an illustration for the proposed restoration process. Let there be two m-trails $T_1 : c-b-a-e-f$ and $T_2 : b-a-e-f$ which help nodes a, b, c, e, and f to localize the failure at link $b-c$. With this, node a can identify the failure at $b-c$ by taking time t_{nc}^{b-c} right after the failure of $b-c$, and node a looks up its RT and immediately switches over the traffic by taking another latency of t_d. Meanwhile, nodes e, f, and c can also identify the failure by taking propagation time t_{nc}^{b-a-e}, $t_{nc}^{b-a-e-f}$, and $t_{nc}^{b-a-e-f-d}$, respectively, and they look up their RTs and immediately configure their OXCs to realize the P-LP which takes t_d. Thus, the total consumed time is the longest of the propagation times plus a single device configuration time.

4.3.1 An Example on the Restoration Process

To achieve NL-UFL, a set of m-trails should be deployed such that each node has an effective ACT for UFL. An illustrative example is shown on Fig. 4.3(a) with all single link failures and one double-link SRLG of $(0,1)$ and $(2,3)$. In this case, all the four nodes can individually perform L-UFL for any SRLG in $\mathcal{Z}$ by inspecting the locally available on-off status of the traversing m-trails. For example, node 1 can achieve local UFL by keeping the ACT as shown in Fig. 4.3(a), in which the on-off status of the traversing m-trails T_1, T_2, and T_3 can be used to uniquely identify any SRLG failure via the look-up-table of the ACT. Each row of the ACT corresponds to a failure state. For example, if node 1 finds that T_1 and T_2 become suddenly OFF while T_3 is still ON, the link $(0,1)$ is considered down by matching the first row of ACT; and if T_1 and T_2 are ON while T_3 is OFF as shown in the third row of the ACT, link $(2,3)$ is considered down. Similarly, node 0, 2, and 3 can locally perform UFL by maintaining their respective ACT, each of which keeps the mapping between all the considered failure states and the on-off status of the traversing m-trails.

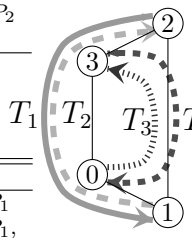
With NL-UFL, the proposed failure restoration process can be performed locally in the optical domain without relying on any multi-hop signaling. To achieve this, each node should maintain a *restoration table* (RT) extended from its local ACT as shown in Fig. 4.3. Compared to ACT, an RT has one more column which keeps the mapping from every localized failure state (i.e., the obtained alarm code) to the corresponding reconfiguration on the OXC. The nodes along an P-LP, including the switching node,

Restoration Table at node 3

ACT_3	T_1	T_3	T_4	OXC Configuration
$(0,1)$	0	1	1	P_1 is configured
$(1,2)$	0	0	1	P_1 is configured, W_2 is switched to P_2
$(2,3)$	1	0	1	W_2 is switched to P_2
$(3,0)$	0	1	0	
$(0,1)$& $(2,3)$	1	1	1	

Restoration Table at node 2

ACT_2	T_1	T_3	T_4	OXC Configuration
$(0,1)$	1	1	0	W_1 is switched to P_1
$(1,2)$	0	1	0	W_1 is switched to P_1, W_2 is released
$(2,3)$	1	0	1	W_2 is released
$(3,0)$	1	0	0	
$(0,1)$& $(2,3)$	1	1	1	



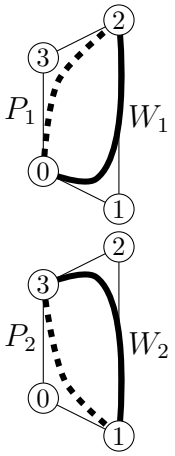
Restoration Table at node 0

ACT_0	T_1	T_3	T_4	OXC Configuration
$(0,1)$	0	1	1	W_1 is switched to P_1
$(1,2)$	0	0	1	W_1 is switched to P_1, P_2 is configured
$(2,3)$	1	0	1	P_2 is configured
$(3,0)$	1	0	0	
$(0,1)$& $(2,3)$	1	1	1	

Restoration Table at node 1

ACT_1	T_1	T_3	T_4	OXC Configuration
$(0,1)$	1	1	0	W_1 is released
$(1,2)$	0	0	1	W_1 is released, W_2 is switched to P_2
$(2,3)$	1	0	1	W_2 is switched to P_2
$(3,0)$	1	0	0	
$(0,1)$& $(2,3)$	1	1	1	

(a) An NL-UFL m-trail solution



(b) Connections

Figure 4.3: The restoration table (RT) at each node is extended from the ACT by having the right-most column which demonstrates the required operations corresponding to each identified alarm code.

intermediate nodes, and the merge node, can start configuring their OXCs to form the required cross-connect by looking up their RTs once a valid alarm code is obtained. With this, the P-LP can be formed right after the identification of the failure event without any further path setup mechanism. In the same example, let two connections, denoted as W_1 and W_2, be launched in the network, which are protected by P_1 and P_2 under any single link failure, respectively, as shown in Fig. 4.3(b). In this case, node 1 should identify whether link $(1,2)$ or $(0,1)$ is failed in order to protect W_1, and if so it should switch over W_1 to P_1.

Note that the RT of a node should be updated when any connection whose P-LP traverses through the node is being set up or torn down.

4.4 The Spare Capacity Allocation (SCA) Problem

In this section we define the SCA problem, the interested reader can refer to [75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89] for more description on the heuristic algorithms solving the SCA problem.

4.4.1 The FDP-SCA Problem Formulation

A key issue to achieve the proposed FDP restoration framework is an approach to determine how m-trails and P-LPs under FDP are allocated, when we are provided with the network topology and a set of W-LPs. The spare capacity problem is different from any previously reported problems since it considers the allocation of both monitoring resources (for m-trails) and restoration resources (for P-LPs). The input of the problem is as follows.

1. The *network topology*, which is represented by an undirected graph $G = (V, E)$. For the sake of simplicity we assume infinite capacity on each link.

2. The *set of SRLGs*, which is denoted by $\mathcal{Z}$. Each SRLG $z \in \mathcal{Z}$ contains one, or at most two and adjacent links.

3. A set of W-LPs denoted as $\mathcal{W} = W_1, W_2, \ldots, W_k$, where k is the total number of W-LPs. Each W_j is a path in G between nodes s_j and d_j for $j = 1, \ldots, k$.

The SCA problem is to minimize the total spare capacity (denoted by $\mathcal{M}$) while achieving: (i) a feasible solution for NL-UFL m-trail allocation under multi-link SRLGs, and (ii) restoration capacity allocation for FDP. Formally

$$\mathcal{M} = \sum_{e \in E} u_e \ , \tag{4.2}$$

where u_e denotes the spare capacity along link $e \in E$ in terms of the number of wavelength links (WLs).

An important feature of the proposed approach is that the spare capacity taken by the m-trails can be reused by any P-LP. This is a reasonable assumption since both spare capacity by P-LPs and m-trails are not launched with working traffic during normal operations. Once there is a failure, all monitoring resources for the m-trails can be released and reused by the P-LPs, thanks to NL-UFL which enables all the nodes to instantly react upon the identified failure. With this, we have:

$$u_e = \max\{m_e, p_e\},$$

where m_e denotes the required monitoring resources for the m-trails as

$$m_e = \sum_{1 \le j \le b} I^T_{j,e} \tag{4.3}$$

where $I^T_{j,e}$ is the trail-link indicator (a.k.a global ACT) which is 1 if the jth m-trail passing through link e, and 0 otherwise. And p_e denotes the restoration capacity for FDP on link $e \in E$ that is formalized as follows.

The Restoration Capacity

For each W_j, $1 \le j \le k$, a set of P-LPs denoted as $P^z_j, \forall z \in \mathcal{Z}_j$, is determined such that they are simple paths with source s_j and destination d_j while disjoint from each z. Let $\mathcal{W}^z \subseteq \mathcal{W}$ be a subset of W-LPs, which are the W-LPs possibly interrupted by failure z. Let $\mathcal{Z}_j \subseteq \mathcal{Z}$ be a subset of SRLGs traversed by W_j. The amount of reserved working capacity along link e is denoted as q_e, $\forall e \in E$; formally $q_e = \sum_{1 \le j \le k} I^W_{j,e}$, where $I^W_{j,e}$ is a working path-link indicator which is 1 if W_j passes through link e, and 0 otherwise. Let $I^{P,z}_{j,e}$ be a protection path-link indicator which is 1 if P^z_j traverses e, and 0 otherwise. Let the restoration capacity along link e in the failure event of z be denoted by p^z_e. We have the following relation:

$$p^z_e = \sum_{1 \le j \le k} I^{P,z}_{j,e} - \sum_{W_j \in \mathcal{W}^z} I^W_{j,e}$$

where the term $\sum_{W_j \in \mathcal{W}^z} I^W_{j,e}$ stands for the free capacity gained by *stub release* at link e after the failure event at z which interrupted all $W_j \in \mathcal{W}^z$.

The condition for p_e to restore every affected W-LP $W_j \in \mathcal{W}^z$ by failure z is:

$$p_e \ge \max_{z \in \mathcal{Z}_j} p^z_e \ .$$

In general, the monitoring resource consumption is determined only by the topology and SRLGs considered in the problem regardless of the traffic, which serves as a constant

expense; on the other hand, the amount of restoration resources heavily depends on the amount of working traffic. Thus, when the amount of working capacity increases, more monitoring resources will be reused due to the increased restoration resource consumption, and gradually all the monitoring resources are reused. We define the monitoring resources that are not shared by any P-LP as *monitoring overhead*, denoted by

$$r_e = \max\{0, m_e - p_e\}.$$

The monitoring resources are expected to be completely hidden by P-LPs, provided that we have sufficient working capacity in the network, resulting zero monitoring overhead. This is also referred to as the *monitoring resource hidden property* where all WLs taken by m-trails are also reserved by P-LPs.

4.4.2 FDP Restoration Capacity Allocation

The restoration capacity allocation problem for FDP has been extensively investigated in the past decades, which is formally called *Spare Capacity Placement/Allocation for Path Restoration with stub release*. Detailed descriptions can be found in [75, 78, 76], [90, Chapter 6], [91, Chapter 9.5.2] for both ILP formulations and heuristic approaches. Since we have positioned our study on a signaling-free FDP-based restoration process via all-optical failure localization, we do not focus on novel FDP restoration capacity allocation schemes; instead, we would adopt state-of-the-art FDP spare capacity allocation schemes for the proof of concept and facilitation of performance analysis. To be specific, the ILP in [91] and Successive Survivable Routing (SSR) algorithm in [78] are implemented in Section 4.6 for comparison with other counterparts as well as for understanding the performance behaviors under the monitoring resource hidden property, respectively.

4.5 The Monitoring Resource Hidden Property

With the SCA problem formulated in the previous section, we are particularly interested in the monitoring resource hidden property, which defines and quantifies in what circumstances the monitoring WLs are all reused by the P-LPs. In other words, there is no additional resource consumed due to the deployment of the m-trails and thus the spare capacity consumption is the same as that of the conventional FDP which is optimal among all the protection schemes. As a first study into the proposed framework, we analyze the problem under single-link SRLGs in circulant graphs, aiming to gain deeper understanding on the performance behaviour of the proposed framework.

4.5.1 Lower Bound on the Spare Capacity

Lemma 16. *Let K be a set of links that form a cut in G. The spare capacity along the links in K is at least*

$$\frac{1}{|K|-1}\sum_{e\in K} q_e \leq \sum_{e\in K} p_e \ . \tag{4.4}$$

Proof. In case link e fails, protection routes must be able to circumvent e via the other links in K, thus the spare capacity along those links is at least q_e, formally

$$q_e \leq \sum_{f\in K, e\neq f} p_f \ . \tag{4.5}$$

There are $|K|$ such inequalities for $1 \leq e \leq |K|$. By summing up these inequalities we get

$$\sum_{e \in K} q_e \leq \sum_{e \in K} \sum_{f \in K, e \neq f} p_f = (|K| - 1) \cdot \sum_{e \in K} p_e \ . \tag{4.6}$$

Finally, dividing both sides by $(|K| - 1)$ we get Eq. (4.4). □

Corollary 4. *Let $K^1, \ldots, K^k$ be a pairwise disjoint cuts. Then*

$$\mathcal{M} \geq \sum_{1 \leq l \leq k} \sum_{e \in K^l} \frac{q_e}{|K^l| - 1} \ . \tag{4.7}$$

Corollary 5. *Let $N_G(v)$ denote the set of links adjacent to node v and $\Delta_v = |N_G(v)|$*

$$\mathcal{M} \geq \sum_{v \in V} \sum_{e \in N_G(v)} \frac{q_e}{\Delta_v - 1} \ . \tag{4.8}$$

As a rule of thumb, the minimal ratio of spare to working capacity can be estimated by $1/(\overline{\Delta} - 1)$ [75], where $\overline{\Delta} = 2|E|/|V|$ is the average nodal degree. It was proved for the case when the working capacity is the same on every link [75]. This can be also deduced from Corollary 5 by applying the inequality of arithmetic and harmonic means.

4.5.2 Dominance of Monitoring Resources

With the proposed NL-UFL construction in the $C_n(1,2)$ topologies, we are interested in whether and how much monitoring resources can be hidden by restoration resources. Let the average working capacity per link, the average restoration capacity per link, and average monitoring capacity per link, be denoted by

$$\overline{q} = \frac{1}{|E|} \sum_{e \in E} q_e, \quad \overline{p} = \frac{1}{|E|} \sum_{e \in E} p_e, \quad \overline{m} = \frac{1}{|E|} \sum_{e \in E} m_e,$$

respectively. Formally, the *dominance of monitoring resources* occurs when $\overline{m} \geq \overline{p}$, which serves as a sufficient condition that additional monitoring resources are required by FDP on top of the restoration capacity .

Let θ measure the traffic demand as a percentage of $s-d$ pairs that are loaded with a W-LP. For example, $\theta = 100\%$ means each $s-d$ pair (i.e., $|V|(|V|-1)/2$) is connected by a W-LP. It is clear that with smaller θ, the dominance of monitoring resources is more likely to happen. There naturally comes up an interesting question: with a specific topology, for which values of θ will make the monitoring resources dominant? Is there a lower bound on θ, say 1%, below which $\overline{m} < \overline{p}$ is unconditionally true? In the following theorem we show that there is no such lower bound on θ in a circulant topology $G = C_n(1,2)$ under single-link SRLGs.

Theorem 16. *For any positive $\theta > 0$, there exists a FDP-SCA problem (with topology, SRLG and traffic) where the monitoring resources will never dominate the spare capacity (i.e., $\overline{m} < \overline{p}$).*

Proof. We pick the circulant graphs $G = C_n(1,2)$ as a topology with unit cost along each link, and all single link failures for SRLG. Let $G = C_n(1,2)$ contain a set of nodes denoted as $0, 1, \ldots, n-1$, and edges denoted as $(v, v+1)$ and $(v, v+2)$, for $v = 0, \ldots, n-1$, where

the addition is understood modulo n. Let us call the edges $(v, v+1)$ by on-cycle edges and the rest chordal edges (see also Fig. 2.10). As for the traffic, let G be launched with a set of shortest-path routed W-LPs denoted as $\mathcal{W} = W_1, \ldots, W_k$, such that $k \leq \theta \cdot \frac{n(n-1)}{2}$. Let $h = \max\{5, \lceil \frac{1}{\theta} \rceil\}$ and $n = 4h \geq 20$. Nodes s and d are connected with a W-LP along the shortest path route if $s - d \equiv 0 \mod h$. In this case the number of directed connections is $k = \frac{3}{2}n$, because each source node s is connected with $s + h$, $s + 2h$ and $s + 3h$, where addition is understood modulo n. Therefore, the total number of directed connections is

$$\theta \cdot \frac{n(n-1)}{2} = \theta \cdot 4 \cdot \max\{5, \lceil \frac{1}{\theta} \rceil\} \frac{n-1}{2} \geq 2(n-1) \geq \frac{3}{2}n = k$$

Next, we define a set of disjoint cuts $K^1, \ldots, K^{n/4}$, where K^i contains 6 links $(v-1, v+1)$, $(v, v+1)$, $(v, v+2)$ for both $v = 2i$ and $v = 2i + n/2$. Each cut K^i, $i = 1, \ldots n/4$ separates the graph into two equal-size fragments with $\frac{n}{2}$ nodes, thus the number of W-LPs passing through the cut is at least n, because for each source node s the number of possible destination nodes in the other side of the cut is 2, thus there are at least n for which $s - d \equiv 0 \mod h$ holds.

According to Corollary 4 we have $\mathcal{M} \geq \frac{1}{5} \cdot n \cdot \frac{6}{4}n$, where $\frac{6}{4}n$ is the total number of links in the cuts $K^1, \ldots, K^{n/4}$. Since the number of links is $|E| = 2n$ we have $\overline{p} = \frac{\mathcal{M}}{2n}$. According to Theorem 7, the total monitoring capacity of the bm-trails is $n\lceil \log_2 n + 1 \rceil + n$. Thus we have $\overline{m} = \frac{1}{2}(\lceil \log_2 n + 1 \rceil + 1)$. Therefore, $\overline{m} < \overline{p}$ holds when $\lceil \log_2 n + 1 \rceil + 1 < \frac{3}{10} \cdot n$ which is always true for $n \geq 20$. □

It is important to note that although necessary, the dominance of monitoring resources is not sufficient for nonzero monitoring overhead.

4.6 Performance Evaluation

4.6.1 Comparison of Signaling-Free Protection Methods

First, a case study is showed to examine the proposed framework in terms of (1) capacity efficiency, (2) restoration time, and (3) computation time. We launched the m-trails to achieve NL-UFL and implemented a FDP scheme (referred to as "FDP") as a realization of the proposed framework, which is compared with a couple of signaling-free restoration schcmes, namely Collaborative Failure Protection (referred to as CFP) and p-Cycle. Both p-cycle and CFP organizes spare capacity into pre-cross-connected cycles to achieve signaling free protection. CFP can achieve better capacity efficiency by reusing the released working capacity of the disrupted lightpaths (i.e., stubs). It is achieved in a cooperative manner among the failure-aware nodes. We computed the optimal solutions by implementing and solving the ILP for each of the three schemes, specifically the one in [73], [92] and [91, Chapter 9.5.2] for p-Cycle, CFP, and FDP, respectively, in which the same link cost and traffic values were adopted to ensure the experiment environments completely in line with the previous art. We assume the given traffic matrix corresponds to 1 Erlang; each link has 16 WLs, and the W-LPs are shortest-path routed in each case. The results were obtained by solving the ILPs using CPLEX v.11, all with a zero gap to the optimal. Smallnet [92] topology (10 nodes, 22 edges) is adopted.

Capacity Efficiency

Fig. 4.4 compares the capacity efficiency of the three schemes. It clearly shows that the performance of the proposed FDP is dominated by the monitoring resource consump-

tion by the m-trails when the traffic load is slight, but becomes the most efficient one when the traffic load is increased because all the monitoring resource consumption can be hidden by the spare capacity taken by the P-LPs of the FDP scheme. Note that PDF yields the optimal capacity efficiency among all the protection schemes, and the capacity consumption by FDP is completely dominated by the PDF spare capacity when the traffic load is heavy enough to hide all the monitoring resource consumption.

Restoration Time

We assume the average physical length of a link is 200km, which requires 1ms for light to traverse; thus the failure localization and notification delays under FDP is $t_{nl} < 20$ms since the longest m-trail is no longer than 20 hops. By assuming $t_p + t_c = 10$ ms and $t_d = 20$ms, this leads to an overall average restoration time of 50ms. For CFP and p-Cycle, the maximal length of each cycle is 10 hops, which yields $t_l < 10$ms for failure localization and notification; however we also have $t_d = 20$ms required at the two nodes for switch fabric configuration and $t_p + t_c = 10$, which results in 40ms in total. For CFP, the longest p-Cycle has 7 links, which leads to slightly shorter restoration time than 40 ms. Under the specification of the case study, the three schemes lead to very similar restoration time.

Computation Time

The computation time was $39,004$sec, 17.22sec, and 3.94sec for CFP, p-Cycle, and FDP, respectively. The proposed FDP has the best computation efficiency.

Under Multi-Link SRLGs

We have adopted the problem instance in [74] for comparison, which is, to the best of our knowledge, the only previously reported study that provides an ILP for static multi-link SRLG p-cycle design. Note that all the other studies for multi-link SRLGs using p-cycles have focused on reconfiguration, rerouting, and dynamic reconfiguration of spare capacity, which do not fit into the targeted scenario. The NSF network (14 nodes, 22 links) topology is used for all single-link and double-link SRLGs. P-cycle requires 965 bandwidth unit while NL-UFL&FDP 302. This clearly shows that the proposed NL-UFL&FDP approach can gain even more advantage when multi-link SRLGs are considered. The average length of m-trails is 20 links, while the maximum is 26 links, which still results sub 50ms restoration time for every single and double failure with NL-UFL&FDP approach.

4.6.2 Monitoring Resources Hidden

We have seen that the proposed approach is outperformed by its counterparts when the working traffic load is small, but will become much more efficient when the number of W-LPs increases, and the monitoring resources can be completely hidden by the restoration capacity for FDP. Therefore, we claim that the proposed approach can achieve the same capacity efficiency as conventional FDP provided the network is loaded with sufficiently large working capacity. This subsection provides extensive simulation results in a wide range of network topologies and SRLG densities, so as to gain deeper understanding on the monitoring resource hidden property under the proposed restoration framework.

The randomly generated planar graphs are classified according to parameter g, which is the length of the longest inner face contained in the graph. Clearly, graphs with smaller values of g are considered more densely meshed; and in the generated graphs, the average nodal degree of each graph ranges from 5.4 (for $g = 3$) to 2.76 (for $g = 7$).

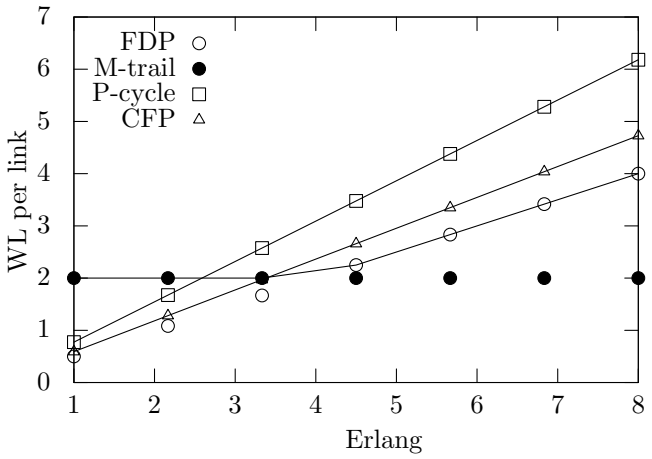
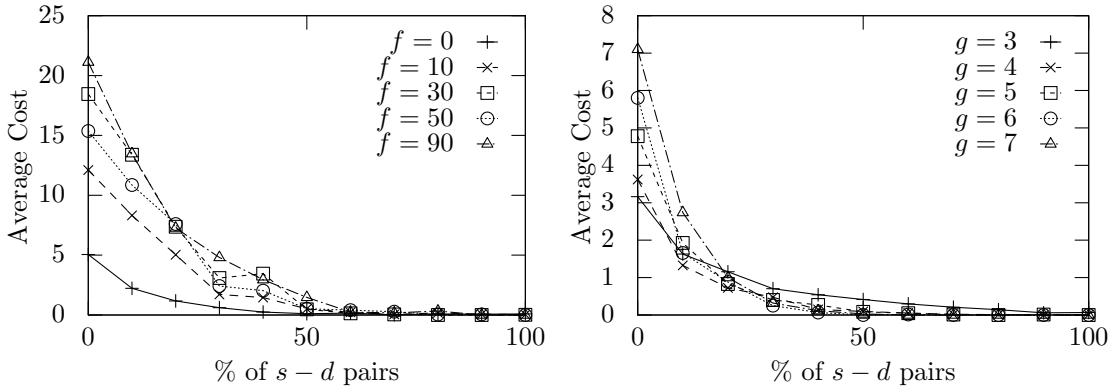


Figure 4.4: WLs fper link on Smallnet.



(a) COST266 network (b) Random networks with $f = 0$

Figure 4.5: Average monitoring overhead.

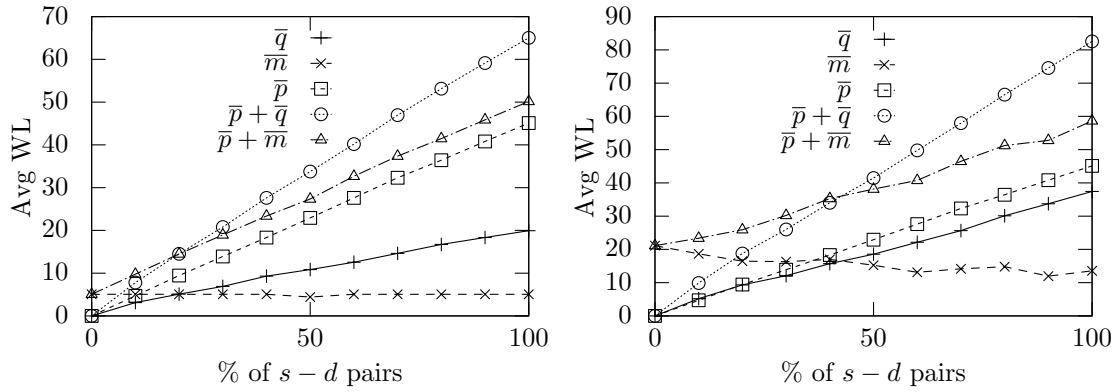
See also Fig. 3.5 for example network topologies with different g. Besides, the COST266 European reference network (37 nodes, 57 links) is also used [93]. We adopted SRLGs with single link and double adjacent links, and the SRLG density considered in the problem is parameterized by a *double failure density parameter*, denoted by f, which indicates the fraction f of all double adjacent link SRLGs under consideration.

We evaluate the resource consumption in the proposed framework by increasing the working traffic, which is defined as the percentage of $s-d$ pairs that are interconnected by a W-LP (i.e., θ). Let every W-LP and m-trail take a single wavelength of bandwidth (i.e., a single WL). The ILP in [91] was no longer used here due to the huge computation complexity; instead, we adopted SSR [78] that sequentially allocates P-LPs for each W-LP. SSR iteratively launches Dijkstra's shortest path algorithm to find disjoint backup paths with each SRLG involved in W-LP one at a time using the latest spare capacity information.

Fig. 4.5(a) and (b) show the average monitoring overhead, i.e.,

$$\frac{1}{|E|} \cdot \sum_{e \in E} \max\{0, m_e - p_e\},$$

as θ is increased from 0% to 100%. It meets our expectation that as the percentage of



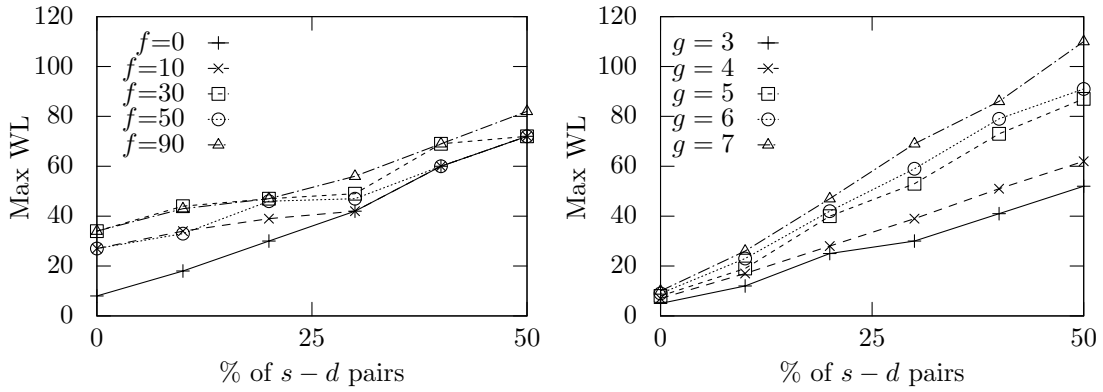
(a) Single-link SRLGs ($f = 0\%$). (b) Adjacent double-link SRLGs ($f = 90\%$).

Figure 4.6: Average used capacity on the links in the COST266 reference network, where working, spare and m-trail capacity is denoted by $\overline{q}$, $\overline{p}$ and $\overline{m}$, respectively.

loaded $s-d$ pairs (i.e., θ) is decreased, and/or as the percentage of double-link adjacent SRLGs (i.e., f) increases, the monitoring overhead increases accordingly. It is interesting to observe that the topology density does not affect the monitoring overhead as shown in Fig. 4.5(b), because taking a sparser topology increases the total monitoring capacity for m-trails, meanwhile increasing the required restoration capacity for FDP, too. As a rule of thumb, we claim that the monitoring overhead is negligible if at least 50% of the node-pairs in a network are loaded with a W-LP. Further, with more than 20% of loaded $s-d$ pairs, the monitoring overhead is 1 WL per link under single-link SRLGs, regardless the network density.

In Fig. 4.6(a) and (b), the average numbers of WLs per link for W-LPs, P-LPs, and m-trails are evaluated in the COST266 European reference network [42] (37 nodes, 57 links) with $f = 0$ and $f = 90\%$ in (a) and (b), respectively. The intersections of the curves for average restoration capacity per link (i.e., $\overline{p}$) and average monitoring capacity per link (i.e., $\overline{m}$) are at $\theta = 20\%$ and $\theta = 40\%$ with an SRLG density $f = 0\%$ and $f = 90\%$, respectively. The curve of $\overline{m}$ shows the contribution adopting in-band information in the reduction of monitoring resource consumption, which is not obvious in the single-link SRLG scenario in Fig. 4.6(a). Nonetheless, the effect of taking in-band information becomes non-trivial when multi-link SRLGs are considered, where $\overline{m}$ decreases from 21.1 to 13.5 as θ increases from 0% to 100%, as shown in Fig. 4.6(b). Further, the reduction of $\overline{m}$ occurs mostly in the light traffic region where the monitoring resources are more likely to be dominant.

Fig. 4.7(a) and (b) show the relation between the maximum total capacity required along each link (i.e., $\overline{q}+\max\{\overline{m},\overline{p}\}$) and θ by using COST266 European reference network (37 nodes, 57 links), under various SRLG densities and topology densities, respectively. In Fig. 4.7(a) when $\theta = 50\%$, the required maximum WLs is roughly 80. This provides us a design guideline that with 80 WLs of total capacity consumed along each link in COST266, the proposed framework can most likely achieve optimal performance, due to the fact that $\theta = 50\%$ is found to be the turning point for reaching zero monitoring overhead under multi-link SRLGs, as shown in Fig. 4.5(a). Further, as shown in Fig. 4.7(b), $30-45$ WLs of overall capacity along each link is the threshold for zero monitoring overhead for single-link failures, because $\theta = 20\%$ is the turning point of zero monitoring overhead



(a) COST266 network (b) Random networks with $f = 0$

Figure 4.7: Maximum overall capacity in WLs.

under single-link SRLGs, as shown in Fig. 4.5(b).

Chapter 5

IP Fast ReRoute

5.1 Introduction

Transporting delay and loss sensitive traffic in the Internet has become an important requirement in the last few years. At the moment the IP protocol suite is not yet ready to fully support multimedia streams due to various reasons, one of which is slow response to failures. Recovery with current Interior Gateway Protocols (IGPs) is in the order of hundreds of milliseconds [94], typically beyond what is tolerable to a multimedia stream. Similar is the case of MPLS networks that rely on LDP for label exchange, as LDP is dependent on the IGP for routing. Therefore, the IETF defined a framework called IP Fast ReRoute (IPFRR [95]), for native IP protection in order to reduce failure reaction time to tens of milliseconds in an intra-domain, unicast setting.

IPFRR is based on two principles: *local rerouting* and *precomputed detours*. Local rerouting means that only routers directly adjacent to a failure are notified of it, which eliminates one of the most time-consuming steps of IGP-based restoration: global flooding of failure information. Additionally, IPFRR mechanisms are proactive in that detours are computed, and installed in the forwarding engine, long before any failure occurs. Thus, when a failure eventually shows up, routers are able to switch to an alternate path instantly. Once alternate next-hops are active, traffic flows undisrupted bypassing the failed component, letting the IGP to converge in the background.

This chapter deals with two specifications for IPFRR, namely the Loop-Free Alternates (LFA) and Protection Routing (PR). The former is the only standardized and readily available IPFRR technology, while the latter is a more theoretic framework as it requires a centralized unit that controls the forwarding table at each router but it still can be implemented with OpenFlow routers.

5.2 Loop Free Alternates

A basic specification for IPFRR is Loop-Free Alternates (LFA, [96]). When connectivity to some next-hop is lost, all traffic that would have used the unreachable next-hop is passed on to an alternate next-hop, called a Loop Free Alternate, that still has a path to the destination that is unaffected by the failure. LFA is simple, it can be realized with straightforward modifications to current IGPs, and deployment is easy thanks to the fact that it does not require support from other routers. On the other hand, LFAs do not always protect both link and node failures at the same time and may also lead to temporary loops when multiple simultaneous failures show up. But the major problem is

that often not all routers have LFAs to all other routers, which means that certain failure patterns are impossible repair rapidly.

Unfortunately, complete IP-level local protection is difficult due to the IP's destination-based forwarding paradigm. As only adjacent routers are aware of a failure, remote routers do not know whether an arriving packet is traveling on its shortest path or it is already on a detour and so exceptional forwarding should be applied. Without being able to differentiate between these two cases, local IP protection can never attain 100% failure coverage[1]. Most IPFRR proposals, therefore, either change IP's destination-based forwarding [97, 98, 99] or introduce some forms of signaling to indicate that a packet is on a detour. Some call for out-of-band failure signaling [100], others use invaluable extra bits in the IP header [101, 102] or add special information to it for in-band signaling [103], and still others propose to mark detours by tunneling [104, 105, 106]. While modern routers can handle tunneled packets at wire speed, tunneling needs additional address management [107, 106] and, if the additional IP header does not fit into the MTU, can cause packet fragmentation and time-consuming reassembly at the tunnel endpoint. It seems, therefore, that the price for IP-level local protection, capable to handle all possible failure cases, is considerable added complexity and management burden, modifications to the essential IP infrastructure, and the breaking of the incremental deployment path.

It is, therefore, no wonder that today LFA is the only standardized and readily available IPFRR technology. At least two major router vendors are offering LFA-based IPFRR support out of the box [108, 109], and other vendors are expected to follow suit. Consequently, operators in need for improving network resilience are now facing the question whether to deploy LFA, and this decision depends crucially on the extent to which LFA can protect failures in the particular topology. We intend to assist making this decision.

Even though thorough, simulation-based reports are available [110, 111, 112, 113, 114], a deep understanding of how certain network characteristics affect LFA failure coverage is still missing. Thus, in this section our focus is on the graph topological ingredients needed for good LFA protection. In this regard, our work is a sequel to [115], where the authors study to what extent IP's destination-based forwarding permits protection routing, and [116] presenting a similar study for the O2 scheme. As we find that LFA failure coverage strongly depends on the topology as well as on the link costs, we study the effects of both separately.

Existing proposals modify standard IP forwarding in some way to achieve full protection. Why not choose the other way around? That is, instead of bending IP to provide full protection in all networks, paying huge price in complexity and deployability, why not bend the network topology itself so that even LFA can guarantee full protection? We study this question in the first part of the chapter. We show real networks where by adding just two or three new links full LFA protection can be attained. This might be an acceptable price for an operator to take the easy deployment path. In some cases, however, our analysis reveals that full LFA protection can only be achieved at the cost of a substantial topology redesign, which is a clear indication to choose alternative protection schemes [81]. At the least, such an analysis can be instructive in the next regular network upgrade cycle.

[1]One can easily prove this claim formally, taking the example of a ring with at least 4 nodes.

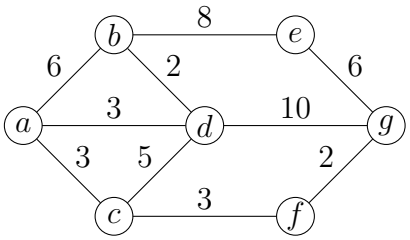


Figure 5.1: A sample weighted network topology.

5.2.1 An Example on Loop Free Alternates

We demonstrate LFA through an example. Consider the network depicted in Fig. 5.1 and suppose that router d wishes to send a packet to router f. The next-hop of d along the shortest path towards f starts for c. If, however, link (d, c) fails, then d needs to find an alternative neighbor to pass on the packet. It cannot send the packet to, say, b, as b's shortest path to f goes through itself, so b would send the packet back causing a loop (remember that in IPFRR, routers not immediately adjacent to a failure do not get notified of it). Instead, it needs to find a neighbor that is closer to the destination than the length of the route from the neighbor through itself. This relation can be expressed as follows:

$$\text{dist}(n, d) < \text{dist}(n, s) + \text{dist}(s, d) \ , \tag{5.1}$$

where s is the actual source node, d is the node the packet is destined to, n is a neighbor of s other than the failed next-hop and $\text{dist}(x, y)$ denotes the length of the shortest $x \to y$ path. A neighbor fulfilling (5.1) is called a *link-protecting Loop Free Alternate* (LFA). For instance, a is a link-protecting LFA for node d towards node f as $\text{dist}(a, f) < \text{dist}(a, d) + \text{dist}(d, f)$.

Many alternative LFA types exist. For instance, g is also an LFA for d towards f, but it also protects against the failure of node c, so it is a *node-protecting LFA*. It is also a *downstream neighbor*, as it is closer to f than d, as well as a *per-link LFA* for the link (d, c) as it protects all the nodes reachable by d through (d, c). For a full taxonomy, see [96, 113, 114].

5.2.2 Model

We model the network topology by a *simple, undirected weighted graph* $G = (V, E)$. Let $\overline{E}$ denote the complement of the edge set. Let the cost of edge (i, j) be $c(i, j)$. For simplicity, we assume that *(i)* edges are bidirectional and point-to-point; *(ii)* costs are symmetric; and *(iii)* each node has a well-defined next-hop towards each destination. This means that if multiple shortest paths are available to a destination, then one is chosen arbitrarily.

In this section we limit our attention to *single link failures* exclusively. As simple *link-protecting LFAs* safeguard against just this type of failures and they contain all the other LFA types as special cases, we do not treat those henceforth. Consequently, we shall usually assume that the graph describing the network is *2-edge-connected*, which is the minimum topological requirement for being able to repair every possible link failure.

5.2.3 Problem Definition

Definition 2. *Consider an undirected, weighted graph $G = (V, E)$. For each $d \in V$, define a relation $(\prec_d)$ on V as follows: let $u \prec_d v$ if at least one shortest path from v to d goes through u. Let $u \preceq_d v$ if either $u \prec_d v$ or $u = v$. Finally, put $u \not\succcurlyeq_d v$ if u and v are not ordered by $(\preceq_d)$.*

In Fig. 5.1, for instance, $d \preceq_f b$ but $d \succeq_b b$, and $a \not\succcurlyeq_f d$.

The relation $(\preceq_d)$ defines a partial order on V, since it is reflexive, transitive and antisymmetric. The partially ordered set $(V, \preceq_d)$ is called the d-poset. Each d-poset has exactly one lower bound: d. We say that some $u \in V$ is an ancestor (descendant) of some $v \in V$ in the d-poset if $u \prec_d v$ ($u \succ_d v$, respectively). Additionally, a parent (child) is a neighboring ancestor (descendant). By assumption, if a node has multiple parents, then one is assigned as next-hop arbitrarily.

Using this model, we redefine (5.1) as follows:

Definition 3. *For some $s \in V$ and $d \in V$, a neighbor $n \in V$ of s that is not the next-hop is a link-protecting LFA (simply LFA, henceforth) if $s \not\preceq_d n$.*

Simply put, n is an LFA if no shortest path from n to d passes through s. Hence, no loop will arise if a packet destined to d is handed by s to n instead of its primary next-hop. Note that the condition $s \not\preceq_d n$ means that either $s \not\succcurlyeq_d n$ or $s \succ_d n$.

From the above discussion, it is clear that in general networks not all nodes have LFA protection to every other node [110, 111, 112, 113, 114]. To measure the *LFA failure coverage* $\eta(G)$ in a weighted graph G, we adopt the simple metric from [96]:

$$\eta(G) = \frac{\#\text{LFA protected } (s,d) \text{ pairs}}{\#\text{all } (s,d) \text{ pairs}}$$

5.2.4 Bounds on LFA coverage

Next, we present simple lower and upper bounds on LFA coverage. Our bounds are based on the following idea. The shortest path tree to some destination d can contain only $n-1$ edges, and all the remaining edges can be used for providing LFAs to their endpoints. In particular, an out-of-tree edge provides at most 2 LFAs (either node-protecting or link-protecting or both), and at least 1 link-protecting LFA towards d.

Let Δ denote the average node degree in G and let $\Delta_{\max}$ be the maximum degree. It is immediate that, $\Delta \geq \frac{2(|V|-1)}{|V|}$ for any connected graph, since the sparsest connected graphs are trees for which $\Delta = \frac{2(|V|-1)}{|V|}$. A Δ-regular graph is a graph in which all nodes are of constant degree Δ. An even (odd) ring is a cycle graph with an even (odd) number of nodes. Rings are the smallest-degree 2-connected regular graphs (in particular, $\Delta = 2$).

In [12] the following lemma was proved

Lemma 17. *For an even ring with $|V| > 2$ and uniform costs: $\eta(G) = \frac{1}{|V|-1}$. For an odd ring with $|V| > 2$ and uniform costs: $\eta(G) = \frac{2}{|V|-1}$.*

Theorem 17. *For any connected simple graph G with $|V| > 2$:*

$$\eta(G) \leq \frac{|V|}{|V|-1}(\Delta - 2) + \frac{2}{|V|-1} \ .$$

Proof. We observe that an edge not contained in the shortest path tree rooted at some d provides at most 2 link-protecting LFAs towards d (when the edge lies between two branches of the tree), while on-tree edges do not create any LFA. Since the number of out-of-tree edges is exactly $|E|-(|V|-1)$, at most $2(|E|-|V|+1) = |V|\Delta - 2|V| + 2 = |V|(\Delta-2)+2$ nodes can have LFA to d. Taken the sum over all nodes and dividing by the number of source-destination pairs gives

$$\eta(G) \leq \frac{|V|(|V|(\Delta-2)+2)}{|V|(|V|-1)} = \frac{|V|}{|V|-1}(\Delta-2) + \frac{2}{|V|-1}.$$

□

The Lemma is non-trivial for $\frac{2(|V|-1)}{|V|} \leq \Delta < 3$. For trees, in particular, we obtain $\eta(G) \leq 0$, which implies that the Lemma is tight for trees over arbitrary link costs. It is tight for uniform cost odd rings as well, for which we obtain $\eta(G) \leq \frac{2}{|V|-1}$ according to Lemma 17.

What the above Lemma in essence says is that in large sparse graphs LFA-coverage is upper bounded by the average node degree: $\eta(G,c) \leq \Delta - 2$. In the course of our numerical evaluations, we found that this relation is present in most real-world networks as well (see later).

The next Lemma gives a lower bound on the LFA coverage. Note, however, that the result concerns link-protecting LFAs exclusively.

Theorem 18. *For any connected simple graph G with $|V| > 2$:*

$$\eta(G) \geq \frac{|V|}{|V|-1}\frac{\frac{\Delta}{2}-1}{\Delta_{\max}-1} + \frac{1}{(|V|-1)(\Delta_{\max}-1)} \ .$$

Proof. Again, exactly $|V|-1$ nodes are contained in the shortest path tree of d, and an out-of-tree edge (of which we have $|E|-|V|+1$) can provide at least one LFA towards d: *(i)* if the edge is inside a single branch of the shortest path tree, then it provides LFA from the upstream to the downstream; *(ii)* if the edge lies between two branches, it still creates at least one link-protecting LFA (in fact, it creates two), but it might not create any node-protecting LFA at all. In consequence, there are $|E|-|V|+1$ out-of-tree edges that are incident to at least $\frac{|E|-|V|+1}{\Delta_{\max}-1} = \frac{|V|(\frac{\Delta}{2}-1)+1}{\Delta_{\max}-1}$ nodes providing a link-protecting LFA towards d (the term $\Delta_{\max}-1$ is because every node has at least one in-tree edge, so only the rest count as out-of-tree edges). Taking the sum over all nodes and dividing by $|V|(|V|-1)$ gives the required result. □

The most important message here is that LFA coverage increases with Δ, that is, the denser the network the higher the link-protecting LFA coverage.

Corollary 6. *For a Δ-regular graph R_Δ on $|V|$ nodes:*

$$\eta(R_\Delta) \geq \frac{1}{2} - \frac{1}{2}\frac{|V|-\Delta-1}{(|V|-1)(\Delta-1)} \ .$$

This gives $\eta(R_2) \geq \frac{1}{|V|-1}$ and $\eta(R_3) \geq \frac{1}{4} + \frac{3}{4}\frac{1}{|V|-1} > \frac{1}{4}$. From this, we conclude that the lower bound of Lemma 18 is tight for even rings. One easily sees that it is tight for trees as well.

We have seen that LFA coverage fundamentally depends on the average node degree. This raises the question whether we can find graphs of low degree with 100% LFA coverage. We found that the 2-connected graph with the smallest possible average degree that can still be fully protected using LFA is the 3-ring C_3. Every other 2-connected graph with complete LFA coverage has average degree higher than 2. By Lemma 17, $\eta(C_3, c) = 1$ which is attained when c is uniform, and one easily sees that $\eta(C_3, c)$ is the only 2-connected graph of average degree $\Delta = 2$ with this property. Graphs with $\Delta < 2$ cannot have full protection because such graphs contain at least one node with degree 1 whose single outgoing link can never be protected. On the other hand, larger 2-connected graphs with $\Delta = 2$ are all ring topologies, and rings can only have full LFA coverage if $n = 3$ (again, by Lemma 17).

5.2.5 The LFA Topology Extension Problem

The conditions for full LFA coverage turn out to be somewhat restrictive, suggesting that only special topologies admit full protection. Indeed, practical studies show that in common networks LFA coverage is usually in the order of 50-90% [110, 111, 112, 113, 114]. There are essentially three approaches to increasing LFA coverage: changing link costs [14, 117], changing the topology [12, 13], or doing both.

Next we focus on the question how to extend the network with new links (e.g., by leasing additional capacity, provisioning new virtual links, or deploying new fibers) to improve LFA coverage, and we shall recur to manipulating costs only if improvement cannot be achieved otherwise. The practical reason behind it is that in many operational networks edge costs are deliberately optimized for the purposes of load-balancing, reducing delay and improving resiliency [118], or to obviate equal-cost paths in order to eliminate packet re-ordering, unwanted packet fragmentation [119], etc. Modifying costs would destroy carefully engineered shortest paths and this would make deploying LFA less attractive to operators.

We study the impacts of the graph topology and edge costs separately.

LFA Graph Extension: Uniform Link Costs

In this section, we ask how to add edges to a graph to achieve full LFA coverage, provided that both the edges originally existing in the graph and the edges we add have the same cost. Obviously, we want to do this with the fewest new edges possible. Consider the problem statement:

Definition 4. LFA graph extension problem in the uniform cost case (minLFAu): *Given a simple, undirected graph $G = (V, E)$ with uniform edge costs c on all edges and an integer l, is there a set $F \subseteq \overline{E}$ with $|F| \leq l$ and $\forall (u, v) \in F : c(u, v) = c$ so that for $G' = (V, E \cup F)$ we have $\eta(G') = 1$?*

Next, we turn to the characterization of networks with perfect LFA coverage. Herein, we concentrate on the uniform cost case, when shortest path routing boils down to min-hop routing.

Observation 1. *Consider an undirected graph G with uniform edge costs. Now, $\eta(G) = 1$, if and only if each node has an LFA towards each of its neighbors.*

Easily, if all neighbors are protected, then all nodes in the graph are protected as well since these are reached through the neighbors. The other way around: if $\eta(G) = 1$ then,

evidently, all neighbors must be protected. We will use the following theorem taken from [12].

Lemma 18. *Consider an undirected, simple graph $G = (V, E)$ with uniform costs. Now, $\eta(G) = 1$, if and only if each edge is contained in at least one triangle (cycle of length 3).*

Proof. First, we show that if all $(u, v) \in E$ are contained in a triangle, then $\eta(G) = 1$. Let some triangle containing (u, v) be $u - v - w$. One easily sees that $u \not\preceq_v w$, as it is the direct path through edge (w, v) that is the shortest (min-hop) path from w to v, and $w \to u \to v$ is strictly longer than that. Thus, w is an LFA for u towards v protecting edge (u, v), and the claim then follows from Observation 1.

To see the reverse direction, we prove that if $\eta(G) = 1$, then every edge is contained in a triangle. If $\eta(G) = 1$, then for each $(u, v) \in E$ node u has an LFA towards v. Let this be w. Easily, $(u, w) \in E$. We only need to show that $(w, v) \in E$ as well to have a triangle. Indirectly, if $(w, v) \notin E$, then $u \preceq_v w$, which contradicts the assumption that w is an LFA. □

Lemma 18 implies that complete graphs, chordal graphs and maximal planar graphs have full LFA coverage in the uniform cost case.

Note that adding uniform cost edges to a uniform cost graph necessarily changes the shortest paths between some of the nodes. This is because at least the nodes connected by the new edge will use it to reach each other, instead of whatever shortest path they had before. Hence, the requirement that shortest paths be invariant to LFA graph extension cannot be met when solving minLFAu.

Theorem 19. *The LFA graph extension problem in the uniform cost case (minLFAu) is NP-complete.*

Definition 5 (SP5, [120])**.** Minimal set cover problem (minSC): *Given a bipartite graph $G' = (A \cup B, C)$ and a positive integer k, is there a set of nodes $B^c \subseteq B$ with $|B^c| \leq k$, such that every node in A has a neighbor in B^c?*

We make the following trivial assumptions: *(i)* each node $a \in A$ is connected to at least two nodes in B (otherwise covering a is trivial); *(ii)* $|B| \geq 3$; and *(iii)* $|A| \geq 2$.

Proof. Instead of solving minLFAu directly, we use Lemma 18. Consider the definition:

Definition 6. Minimal triangular problem (minTR): *Given a graph $G = (V, E)$ and a positive integer l, is it possible to add at most l edges to G so that every edge is contained in a triangle?*

Easily, minTR is solvable for l if and only if minLFAu is also solvable for l. Hence, proving that minTR is NP-complete will yield the required result. MinTR is in NP, since a nondeterministic algorithm needs to guess the set of edges F with $|F| \leq l$ and a polynomial time algorithm can verify if every edge is part of a triangle. To prove that minTR is indeed NP-hard, we (Karp-)reduce minSC to minTR: given a minSC instance with a bipartite graph $G' = (A \cup B, C)$ and an integer k, our task is to define an input graph $G = (V, E)$ for minTR that is solvable with at most k edges, if and only if the minSC instance is solvable with at most k nodes.

Construct $G = (V, E)$ as follows: let $V = A_1 \cup A_2 \cup B \cup \{s\}$, $|V| = 2|A| + |B| + 1$. Denote the nodes in V by $a_i^1 \in A_1$, $a_i^2 \in A_2$, $b_j \in B$ and s, respectively, where $i = 1, \ldots, |A|$ and $j = 1, \ldots, |B|$. Additionally, let $E = E_1 \cup E_2 \cup E_3$ where:

E_1: (a_i^1, b_j) and (a_i^2, b_j) if $(a_i, b_j) \in C$,

E_2: (s, a_i^1) and (s, a_i^2) for $i = 1, \ldots, |A|$,

E_3: (b_j, b_l) for all $j = 1, \ldots, |B|$, $l = 1, \ldots, |B|$ and $j \neq l$.

A minSC instance and its transformation are given in Fig. 5.2.

We say that an edge is protected if it is part of a triangle, unprotected otherwise. We make the following observations: edges in E_1 are protected because of assumption *(i)*, similarly edges in E_3 are also protected because of *(ii)*, while edges in E_2 are all unprotected. The idea is that in order to protect all edges in E_2 we need to add $(s, b_j) : b_j \in B$ edges, called *cover edges* henceforth. Each such (s, b_j) cover edge, when added, protects the $(s, a_i^1), (s, a_i^2) \in E_2$ edges for all $a_i \in A$ nodes adjacent to b_j in G'. Thus, we get a minimal cover of G' exactly when all edges in E_2 become protected.

To conclude the proof we need to show that *(a)* if G' can be covered with k nodes from B, then we can identify k edges to be added to G so that every edge becomes protected; and *(b)* if k edges are added to G so that every edge becomes protected, then we can identify a k node subset of B that covers every node in A.

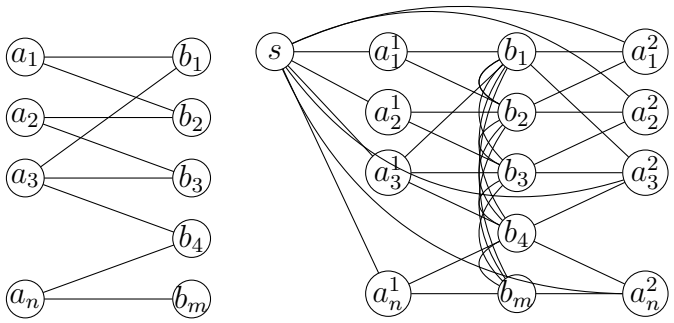
(a) Let $B^c \subseteq B$ be a cover in G' with $|B^c| \leq k$. Add edges $(s, b_l) : \forall b_l \in B^c$ to E. Then, since B^c is adjacent to every node in A_1 and A_2 (due to it being a cover), every edge in E_2 becomes protected. This is because, the edges $(s, a_i^x) \in E_2$, $(a_i^x, b_l) \in E_1$ and (s, b_l) make up a triangle; where a_i^x is adjacent to b_l and $x = 1, 2$.

(b) Suppose that there is a set of cover edges F' such that $F' = (s, b_j) : b_j \in B' \subseteq B$ and B' is adjacent to every node in A_1 and A_2. Then, by the above reasoning, F' can be converted to a set cover. Let $F \in \overline{E}$ be a set of edges which, when added to G, protect all edges. First, we add all cover edges of F to F'. Now, suppose that some edges in F are not cover edges. F does not contain edges of the form $(b_i, b_j) : b_i, b_j \in B$ and $(s, a_i^x) : a_i^x \in A_1 \cup A_2$, since these all exist in E. Edges $(a_i^x, b_j) : a_i^x \in A_1 \cup A_2, b_j \in B$ can be dropped from F, as these do not protect any unprotected edge. What remains are edges of the type $(a_i^x, a_j^z) : a_i^x, a_j^z \in A_1 \cup A_2$. Let us call these *cross edges*. Furthermore, call a node $a_i^x \in A_1 \cup A_2$ protected if the edge (s, a_i^x) is protected. We shall convert cross edges to cover edges before adding to F'. Clearly, we need to consider only those cross edges where both $a_i^1 \in A_1$ and $a_i^2 \in A_2$ are protected by a cross edge, otherwise a cover edge in F' already covers both a_i^1 and a_i^2. Let $A_1' \subseteq A_1$ and the "opposite nodes" $A_2' \subseteq A_2$ be the set of these nodes. Since one cross edge can protect at most two nodes, there are at least $|A_1'|$ cross edges protecting A_1' and A_2', and these are trivially substituted by exactly $|A_1'|$ cover edges: for each $a_i^1 \in A_1'$ choose a neighbor $b_j \in B$ and add the cover edge (s, b_j) to F'. This will protect $a_i^2 \in A_2'$ too. Finally, all nodes B' are adjacent to every node in A_1 and A_2, which completes the proof. □

LFA Graph Extension: Weighted Graphs

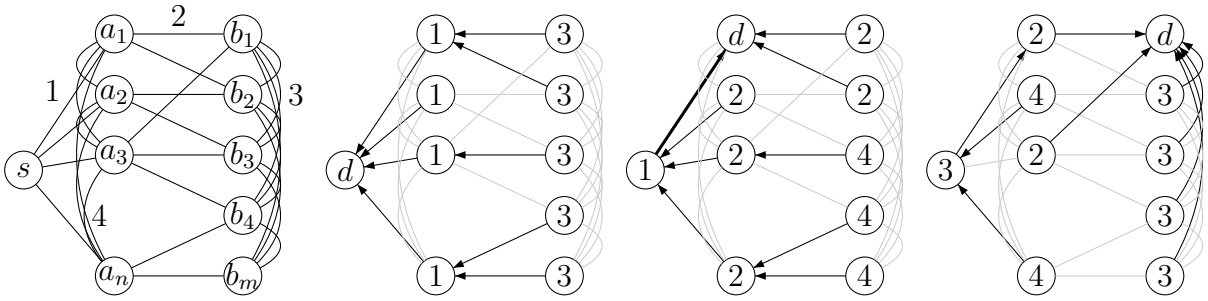
In contrast to uniform cost graphs, where we could not solve the LFA graph extension problem without changing some shortest paths, in weighted graphs we can. If an edge with sufficiently large cost is added to the graph, then shortest paths remain intact while LFA coverage may improve. Here, and in the rest of this chapter, "sufficiently large" will generally mean "larger than the length of the longest shortest path".

Definition 7. LFA graph extension problem in the weighted case (minLFAw): *Given a simple, undirected, weighted graph $G = (V, E)$ and an integer l, is there a set $F \subseteq \overline{E}$ with*



(a) G' (b) G

Figure 5.2: A minSC instance (a) and the converted graph (b).



(a) $G = (V, E)$ (b) $d = s$ (c) $d \in A$ (d) $d \in B$

Figure 5.3: The converted graph topology and link costs. Fig. (a) depicts the graph and Fig. (b), (c) and (d) give the shortest paths and distances with different choices of the destination d.

$|F| \leq l$ and properly chosen costs, so that for $G'(V, E \cup F)$ we have $\eta(G') = 1$ and the shortest paths in G coincide with the shortest paths in G'?

Next we characterize the complexity of minLFAw.

Theorem 20. *The LFA graph extension problem for the weighted case (minLFAw) is NP-complete.*

Proof. The minLFAw problem is in NP, since it was formulated as an ILP in Section 5.2.5. To prove that it is NP-hard, we again reduce minSC to minLFAw (see Definition 5): given a minSC instance with a bipartite graph $G' = (A \cup B, C)$ and an integer k, our task is to define an input graph $G = (V, E)$ for minLFAw, so that minLFAw is solvable by adding at most k edges if and only if minSC is solvable with at most k nodes.

Let us construct $G = (V, E)$ as follows: $V = A \cup B \cup \{s\}$ and $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where

E_1: (s, a_i) with cost 1 for each $i = 1, \ldots, |A|$,

E_2: (a_i, b_j) with cost 2 for each $(a_i, b_j) \in C$,

E_3: (b_j, b_l) with cost 3 for all $j = 1, \ldots, |B|$, $l = 1, \ldots, |B|$ and $j \neq l$,

E_4: (a_i, a_j) with cost 4 for all $i = 1, \ldots, |A|$, $j = 1, \ldots, |A|$ and $i \neq j$.

Fig. 5.3a shows the converted graph for the same minSC instance we used in the previous proof (see Fig. 5.2a).

The idea here is that we embed G' into G and, by carefully choosing the edge costs, we ensure that achieving perfect LFA coverage in G precisely solves the minSC instance on G'. More formally, we show that some $B^c \subseteq B$ is a cover, if and only if $\eta(G'') = 1$ where $G'' = (V, E \cup F)$ for some set of edges $F : \{(s, b) : b \in B^c\}$ of sufficiently large cost. We discuss the LFA coverage for different destinations separately.

1. $d = s$*:* As one easily checks in Fig. 5.3b, each node has an LFA towards destination s. Nodes in A provide LFA for each other as they are not ordered in the d-poset and $|A| \geq 2$ by assumption *(iii)*. Using *(ii)*, one easily shows that nodes in B also provide LFA for each other.

2. $d \in A$*:* All nodes in $A \setminus \{d\}$ have an LFA towards d, since they are not ordered in the d-poset and $|A| \geq 2$ due to *(iii)* (see Fig. 5.3c). Similarly, each $b \in B$ has an LFA from some other node in B. This leaves us with the single node s that does not have an LFA to d. This is because all of its neighbors are in A, but d is its next-hop so it can not be used as an LFA and all other nodes in A are its children. Nodes not connected to d can not provide an LFA to s either, since these are all its descendants. What remains are the neighbors of d. So, we have to add an edge (s, b_j) for some $(d, b_j) \in C$ with sufficiently large cost to protect edge (s, d).

3. $d \in B$*:* First, each $b \in B$ is protected due to *(ii)*. Second, each $a \in A$ that is a neighbor of d is protected by some neighbor $b \in B \setminus \{d\}$. Such neighbor exists due to *(i)*. Similarly, each $a \in A$ that is not a neighbor of d is protected by some neighbor $b \in B \setminus \{d\}$, which again exists due to *(i)*. Again, node s remains without an LFA to d. However, at this point there already is at least one edge $(s, b_j) : b_j \in B$ that we added previously to have an LFA from s to some $a \in A$. This will serve as an LFA for s to d in this case too. Note that it is guaranteed by *(iii)* that at least one such (s, b_j) edge must have been added, and it is enough to have just a single edge from s to B to have an LFA for each $d \in B$.

In summary, in order to have an LFA from s to all nodes in A, we need to add edges from s to some nodes $B^c \subseteq B$. Observe that B^c is adjacent to every node in A, otherwise, we do not have LFA available from s to some of the nodes in A. So B^c is a cover exactly when we have perfect LFA coverage, which completes the proof. □

5.3 Protection Routing

Providing fully distributed, fault tolerant, hop-by-hop routing is one of the key challenges for intra-domain IP networks. This can be achieved by storing two next-hops for each destination node in the forwarding table of the routers, and the packets are forwarded to primary next-hop (PNH), unless PNH is unreachable and secondary next-hop (SNH) is used instead. We follow the architecture by [115], where the routing tables are configured in a centralized way, while the forwarding and failure recovery is in a fully distributed way without relying on any encapsulation and signaling mechanisms for failure notification, to meet the standard IP forwarding paradigm. A network is protected if no single link or node failure results in forwarding loops. Kwong, Gao, Guerin and Zhang [115] conjectured that network node connectivity is not sufficient for a network to be protectable. In the rest of the chapter we show that this conjecture is in contradiction with a conjuncture by Hasunuma [121, 122], and show that every four connected maximal planar graph and every underlying graph of a 2-connected line digraph has feasible protection routing.

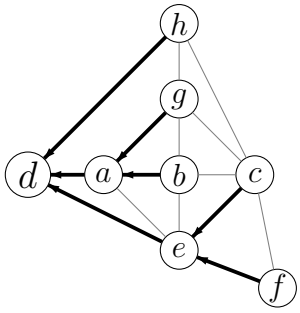


Figure 5.4: A network G with a routing R_d drawn with solid thick arrows.

5.3.1 Motivation

Hop-by-hop routing and IP protocol have become the dominant platform for telecom services [123, 124]. Commercial applications demand reducing the interruption in packet forwarding to sub-50ms in case of failures. As a solution for the problem, in [115] an intra-domain solution was proposed, where a centralized unit pre-computes a primary and an alternate next-hops for each router, where the traffic is instantly switched to the secondary next-hop (SNH) if the primary next-hop (PNH) becomes unavailable. Therefore, forwarding and failure recovery are performed in a fully distributed way relaying on traditional IP forwarding without any encapsulation and signaling mechanisms for failure notification, similarly to O2 [125, 126], DIV-R [127], MARA [128], and LFA [96]. The key challenge is how to avoid forwarding loops in case of failures, when the traffic is forwarded on SNH at a single node. A network is protected if no single link of node failure results in forwarding loops. In this section we investigate how dense should a topology be to become protected.

Kwong, et. al [115] showed that high edge connectivity is not sufficient for a network to be protected against every single link and node failure, and conjectured that high node connectivity is not sufficient either. In this section we investigate the latter issue by applying the results and conjectures of Hasunuma [121, 122] to node connectivity. We show that the conjecture by Kwong, et. al is in contradiction with a conjecture by Hasunuma, besides we define a class of four-node-connected graphs with feasible protection routing.

5.3.2 Problem Formulation

We formulate the protection routing problem at the centralized routing entity. We assume that the network topology information and link bandwidths are available, and the packet forwarding is destination based (hop-by-hop) without reliance on packet marking or encapsulation.

Protection Routing

We model the network as an undirected graph $G = (V, E)$, with V the node set, E the link set. For a destination node $d \in V$ let $R_d = (V, E_d)$ be a *routing* for traffic destined to d, where $E_d \subseteq E$. R_d is a directed acyclic graph (DAG) rooted at d and defines a destination based routing. In R_d every node has at least one outgoing link except for d. Every outgoing link is called *primary link*, and the target node of every primary link is called *Primary Next Hop* (PNH). See also Fig.5.4 as an example.

The routing R_d towards node d can be modeled by a partial order. Define relation $\prec_d$ such that $x \prec_d s$, $x \neq s$ if there exists a path from node s to node x in R_d (i.e. there is a

possible packet flight from s to d through x). Node s is *upstream* of node x if $x \prec_d s$, and conversely, node s is *downstream* of node x if $s \prec_d x$. Additionally, we denote by $x \not\succ_d s$ the case when nodes x and s are not ordered with each other. Besides, the neighboring nodes of node n is denoted by $N_G(n)$. Next let us define link and protection[2].

Definition 8. *A network $G = (V, E)$ is* protected *(with respect to node d) against single* link *failure $f \in E$, if there exists a routing R_d so that either $f \notin R_d$; or $f = (x \leftarrow s) \in R_d$ and node s has a neighboring link $(s, k) \neq f$, such that*

1. *node k is not upstream of node s in R_d (i.e. $s \not\prec k$).*

2. *node k and all its downstream nodes (except d) have at least one PNH in $R_d \setminus f$.*

On Fig. 5.4 every link is protected with the routing. Similarly we define node protection

Definition 9. *The network $G = (V, E)$ is* protected *(with respect to node d) against single* node *failure $f \in V$, if there exists a routing R_d so that every node $s \in N_G(f)$ having primary link to f (i.e. $(f \leftarrow s) \in R_d$) has a neighboring node $k \neq f$ (i.e. $k \in N_G(s)$), such that*

1. *Node k is not upstream of node s in R_d (i.e. $s \not\prec k$).*

2. *Node k and all its downstream nodes (except d) have at least one PNH in $R_d \setminus N_G(f)$.*

In both definitions node k is called Secondary Next Hop and denoted by $SNH_d(s)$. On Fig. 5.4 node e is not protected with the routing, because node f shall forward the packet to node c after the failure of e; however c has a PNH towards the failed node e.

Definition 10. *A routing R_d is* protection routing *in network $G = (V, E)$ with respect to node d if every node and link failure is protected.*

Definition 11. *An undirected graph $G = (V, E)$ is* protectable *if a protection routing exists for all $d \in V$, otherwise the graph is* unprotectable.

Note that G is protectable as it is shown on Fig. 5.6. In 2010 Kwong, et. al has showed [115], that the graph edge-connectivity is not a sufficient condition for a network to be protectable. Besides, for node connectivity, 2- and 3-node-connected unprotectable graphs were constructed, and for networks with higher node-connectivity the following conjecture was given:

Conjecture 1 ([115, Conjecture 4.1])**.** *For any given $k \geq 4$, there exists a k-node-connected graph that is unprotectable.*

In Section 5.3.3. we show that this conjecture is in contradiction with a conjecture by Hasunuma from 2001.

[2]In [115] the two definitions were merged into a more general definition on component failures.

Completely Independent Spanning Trees

Next we define the completely independent spanning trees problem and some general notations. Let T be a spanning tree in graph G. Let $T(s,t)$ denote the path in tree T between nodes s and t.

Definition 12. *Let T^1 and T^2 be two spanning trees of an undirected graph G. We call them* completely independent spanning trees *if for any two nodes s and t the paths from s to d in T^1 and T^2 (i.e. $T^1(s,d)$ and $T^2(s,d)$) are node-disjoint apart from their end nodes.*

In [121] Hasunuma showed that there are k completely independent spanning trees in the underlying graph of a k-connected line digraph and in [122] that there are two completely independent spanning trees in any 4-connected maximal planar graph.

Computing completely independent spanning trees is NP-hard [121]. According to our experiments for real size IP networks it can be done with ILP in reasonable time.

Completely independent spanning trees are special edge-disjoint spanning trees. On edge-disjoint spanning trees, Nash-Williams [49] showed that there are k edge-disjoint spanning trees in any $2k$-edge-connected graph. In 2001 [121, 122] Hasunuma gave a similar conjecture for node-disjoint graphs.

Conjecture 2 ([121], [122, Conjecture 2])**.** *There are k completely independent spanning trees in any $2k$-connected graph.*

The main contribution of this section is to show that two completely independent spanning trees are sufficient for a network to be protected. As a result, both Conjecture 2 and Conjecture 1 cannot be true at the same time.

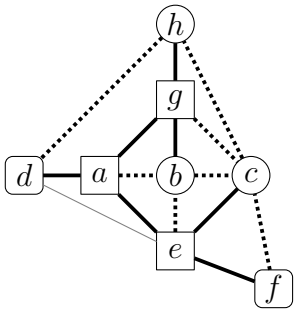
5.3.3 Sufficient Conditions for Protectable Graphs

In routing R_d if a node has multiple outgoing links the traffic is split evenly across them, which is favoured mainly for load balancing issues. As a result, between certain node-pairs the traffic is routed along multiple paths, called Equal-Cost Multi-Path (ECMP). In this section, for the sake of simplicity, we forbid ECMP and consider the case where routing R_d should be a tree instead of a DAG. Clearly, a d rooted tree is a special DAG, and thus with this new constraint the protection routing problem becomes harder to solve. In this case each node n has a single PNH, denoted by $PNH_d(n)$, and R_d is a d rooted spanning tree. An important observation is that when R_d is a spanning tree the second property of Definition 8 and 9 is not needed. In other words our task is to find a d rooted spanning tree R_d, such that each node s has a neighbor node k not upstream to the downstream adjacent node of s, except when the downstream adjacent node is the destination node d.

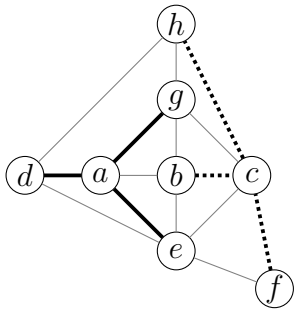
We call a degree one node of a tree a *leaf node*; otherwise it is an *internal node*. In [121] the following property of two completely independent spanning trees was proven.

Theorem 21 ([121, Theorem 2.1])**.** *Let T_1 and T_2 be two spanning trees in the graph G. Then T_1 and T_2 are completely independent if and only if T_1 and T_2 are edge-disjoint and for any vertex v of G there is at most one spanning tree T_i such that node v is an internal node of T_i.*

Fig. 5.5 shows an example of two completely independent spanning trees. According to Theorem 21 the nodes of the graph can be classified into the following three categories. Each node n is either



(a) The circles are leaf nodes of T^1, the boxes are leaf nodes of T^2, and rounded boxes are leaf nodes in both trees.



(b) The two sub-trees: $\hat{T}^1$ is with solid thick lines, and $\hat{T}^2$ is with dashed lines.

Figure 5.5: Two complete independent spanning trees, where T^1 is with solid thick lines, and T^2 is with dashed lines.

1. a leaf node in T_1, or
2. a leaf node in T_2, or
3. a leaf node both in T_1 and T_2.

Next we erase some leaf links from T^1 and T^2 to obtain two sub-trees, called skeletons.

Definition 13. *For each node $n \in G$, we erase a leaf link from either T^1 or T^2. The resulting trees are called* skeletons *denoted by $\hat{T}^1$ for T^1 and $\hat{T}^2$ for T^2.*

Note that according to Theorem 21 a node cannot be an internal node in both trees. Fig. 5.5(b) shows the skeletons $\hat{T}^1$ and $\hat{T}^2$ of the example on Fig. 5.5(a).

Observation 2. *Each skeleton is a connected sub-tree of graph G.*

Proof. $\hat{T}^i$ is a sub-tree of T^i having every internal node and some leaf nodes. □

Observation 3. *Each node $n \in V$ is a part of a skeleton and adjacent to a node in the other skeleton.*

Proof. Completely independent spanning trees T^1 and T^2 cover every node of the graph twice, thus by erasing one (leaf) link form T^1 or T^2 each node remains covered by either $\hat{T}^1$ or $\hat{T}^2$ but not both. By symmetry let us assume that node $n \in \hat{T}^1$. Since node n is a leaf node of T^2, node n is adjacent with an internal node of T^2 which is part of skeleton $\hat{T}^2$. □

Theorem 22. *A graph with two completely independent spanning trees T^1 and T^2 is protectable.*

Proof. We give a deterministic polynomial construction, which builds up a protection routing R_d from two completely independent spanning trees T^1 and T^2 respect to an arbitrary node d. First, we compute the skeletons of the two completely independent spanning trees T^1 and T^2, denoted by $\hat{T}^1$ and $\hat{T}^2$ respectively. By symmetry we assume node $d \in \hat{T}^1$. Let link e be the leaf link of T^2 adjacent to node d. Note that link e connects the two skeletons. The protection routing R_d contains the links of $\hat{T}^1$ and $\hat{T}^2$

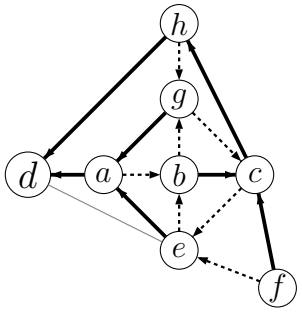


Figure 5.6: Protection routing to node d, where PNH is drawn with solid thick arrows and SNH with dashed arrows.

and link e, and all the links are directed to node d. Fig. 5.6 shows the resulted protection routing of example Fig. 5.5.

To complete the proof we need to show that the network is protected by R_d. Clearly, R_d is a spanning tree because both $\hat{T}^1$ and $\hat{T}^2$ are connected sub-tree of graph G covering every node, and edge e connects the two skeletons. In R_d every node $u \in \hat{T}^1$ is routed to node d along links of skeleton $\hat{T}^1$ only, while each node $v \in \hat{T}^2$ is routed to node d along links of skeleton $\hat{T}^2$ and finally link e. Therefore, the routing from any node in $\hat{T}^1$ and any node in $\hat{T}^2$ are node- and link-disjoint (except for node d). By Observation 3 each node $s \neq d$ is adjacent with a node n form the other skeleton, which can be chosen as SNH, and concludes the proof. □

Note that the proposed construction is much stronger than needed for protection routing, because (1) it guarantees that after a packet is sent to SNH it will travel a fully disjoint route to the destination, compared to its original route. By the definition of protection routing the two routes may be overlapping, and should be disjoint from the failed neighboring link or node. Besides, (2) the proposed construction cannot take advantages of the possible ECMPs.

Corollaries

Corollary 7. *Conjecture 2 by Kwong et al. [115] and Conjecture 1 by Hasunuma [122] is in contradiction.*

Proof. According to Conjecture 1 there are two completely independent spanning trees in any 4-node-connected graph. With two completely independent spanning trees by Theorem 22 a protection routing can be computed, which is in contradiction with Conjecture 2. □

Corollary 8. *Every 4-connected maximal planar graph is protectable, and the protection routing can be calculated in linear time $O(|E| + |V|)$.*

Proof. According to [122, Theorem 2] there are two completely independent spanning trees in any 4-connected maximal planar graph, which can be computed in linear time. With two completely independent spanning trees T^1 and T^2 by Theorem 22 a protection routing can be computed, by iterating through the nodes of the graph. Each step takes linear time. □

Corollary 9. *Every underlying graph of a 2-connected line digraph is protectable.*

Proof. It has been shown in [121] that there are k completely independent spanning trees in the underlying graph of a k-connected line digraph. With two completely independent spanning trees according to the construction of Theorem 22 protection routing can be built. □

Chapter 6

Summary of New Results

In the dissertation we have provided analytical solutions to practical problems improving the reliability of Internet for every user. First I identified some key problems to be solved, and described the formal core to be investigated. To achieve highly reliable Internet solutions, I focused on the following two sub-problems:

Claims 1-2 Optical layer failure localization and restoration. We provided solutions in the backbone networks that could provide much more reliable and flexible service for IP networks. The approach is related to combinatorial group testing.

Claim 3 IP Fast Reroute (IPFRR). IPFRR is designed for fast failure restoration in IP networks. My focus was to improve the performance of Loop Free Alternate (LFA), which is the only commercially available IPFRR in the routers. The related problems are connected to trees and connectivity in graphs, and complexity theory.

6.1 List of Claims

Claim 1. *Failure Localization via a Central Controller*

Claim 1.1. *There is a polynomial time near optimal construction on complete graphs with* $b = 4 + \lceil \log_2 (|E| + 1) \rceil$ *m-trails to achieve UFL for single link failures, when* $|V| \geq 6$ *[5, 8, Theorem 3].*

Claim 1.2. *There is a polynomial time optimal construction on* $2 \cdot \lceil \log_2 (|E| + 1) \rceil$ *connected graphs with* $b = \lceil \log_2 (|E| + 1) \rceil$ *bm-trails to achieve UFL for single link failures, when [129, Theorem 4].*

The theorem is applicable to complete graphs with at least 18 nodes. It improves the $b \approx 2\lceil \log_2 |E| \rceil$ construction by Harvey at. el. [30].

Claim 1.3. *There is a polynomial time near optimal construction on 2D rectangular grid graphs* $S_{m,n}(E, V)$ *with* $b = 3 + \lceil \log_2 (|E| + 1) \rceil$ *bm-trails to achieve UFL for single link failures, when* $m, n \geq 1$ *[6, 129, Theorems 6 and 5].*

It improves the $b \approx 3\lceil \log_2 |E| \rceil$ construction by Harvey at. el. [30]. These constructions are excellent benchmarks as shown in Section 2.2.5.

A circulant $C_n(1, 2)$ graph has nodes $V = \{v_0, v_1, \ldots, v_{n-1}\}$ with each node v_j adjacent to $[v_{j+1} \mod n]$ and $[v_{j+2} \mod n]$.

Claim 1.4. *There is a polynomial time optimal construction on circulant graph $G = C_n(1,2)$ with $b = \lceil \log_2(|E|+1) \rceil$ m-trails to achieve UFL for single link failures. Each m-trail is a spanning sub-graph of G thus it is a near optimal NL-UFL solution with bandwidth cost $\|\mathcal{T}\| = b \cdot n + 1$ [10, Theorem 7 and 11, and Corollary 3].*

Claim 1.5. *To obtain fast and high-quality m-trail solutions, a novel algorithm based on random code assignment (RCA) and random code swapping (RCS) was developed. Through extensive simulations we show that the proposed algorithm can achieve significantly better performance compared to the prior art. The new algorithm uses 3-4 order less computation time without losing solution quality [5, 8] (see also Section 2.2.7).*

Since the appearance of [5] several heuristic algorithms have been developed to solve the same problem. They can achieve slightly better performance however with a significantly longer running time (see also Fig. 2.9).

Claim 1.6. *To obtain m-trail solutions for multiple failures, a novel algorithm based on combinatorial group testing (CGT) and greedy code swapping (GCS) was developed [11]. I have proved that each GCS requires $O(|E|^2 \log |E|)$ steps (see also Section 2.3.2).*

Claim 2. *Distributed Single Failure Localization and Restoration in All-Optical Mesh Networks*

Claim 2.1. *Let $\mathcal{T} = T_1, \ldots, T_b$ be a valid m-trail solution for NL-UFL for single failure. We have the following lower bounds on the bandwidth cost. For sparse networks*

$$\|\mathcal{T}\| \geq \xi |V| \log_2(|E|+1),$$

where ξ depends on the diameter and sparsity of graph [10] (see the details in Theorem 9). And for dense networks

$$\|\mathcal{T}\| \geq 2|E| \left(1 - \frac{1}{|V|}\right)$$

[10, Theorem 12].

In our experiments ξ is somewhere in $[0.85, 0.95]$ for random networks with average nodal degree less than 6. We further improve the lower bound for sparse networks in Theorem 11 for almost $\xi \approx 1.0$. A more general results is proved. Let us consider a generalized combinatorial group testing problem where the cost of a group test T depends on its cardinality $|T|$ according to a given cost function $\omega(|T|)$. Function ω has the following properties:

(i) $\omega(1) = 1$, meaning that a test with one element has a unit cost.

(ii) $\omega(x+1) \geq \omega(x)$ for every positive integer $1 \leq x \leq m-1$. Testing a larger group cannot decrease the cost.

(iii) $\frac{\omega(x)}{x} \geq \frac{\omega(x+1)}{x+1}$ whenever $1 \leq x < m$.

Claim 2.2. *The total cost of localizing a single faulty item among m items with group tests $T_1, \ldots, T_b$ is [7, Theorem 10]*

$$\sum_{i=1}^{b} \omega(|T_i|) \geq \min_{1 \leq x \leq \frac{m}{2}} \omega(x) \left(\log_2 x + \frac{m}{x} - 1\right).$$

Claim 2.3. *There is an optimal NL-UFL construction for single failure on line, complete and star graphs [10, Theorem 13, 14 and 15].*

Inspired by the design of the optimal construction for circulant and complete graphs (Thesis 1.4 and 2.3)a new heuristic algorithm is proposed, which employs spanning tree as the structure for m-trails.

Claim 2.4. *To obtain NL-UFL solutions with bm-trails for single failures, a novel algorithm based on random spanning tree assignment (RSTA) and greedy link swapping (GLS) was developed. The randomly generated spanning trees are deployed into the network at the beginning. Next, to ensure global code uniqueness of each link, GLS resolves possible collisions among preliminarily assigned link codes [10] (see also Section 3.4).*

Extensive simulation on thousands of randomly generated topologies was conducted to verify the proposed heuristic algorithm, where we experienced an average gap of less than 9.7% to the best lower bound.

By integrating NL-UFL failure localization scheme with failure dependent protection (FDP), a novel restoration framework is introduced, which aims to achieve the following three desired features of optical layer protection in all-optical mesh networks:

1. a signaling-free and completely all-optical restoration process;
2. 100% restorability for any multi-link SRLG failure event; and
3. optimal capacity efficiency as FDP.

The key idea is to allow the spare capacity be reused for failure monitoring. Clearly, spare capacity depends on the amount of traffic, while monitoring resources on the topology. Let θ measure the traffic demand as a percentage of $s-d$ pairs that are loaded with a working path.

Claim 2.5. *For any positive $\theta > 0$, there a FDP-SCA problem (with topology, SRLG and traffic) where the monitoring resources will never dominate the spare capacity [Theorem 16].*

Through extensive simulation we showed that the proposed approach achieves zero monitoring overhead over the sparse capacity.

Claim 3. *IP Fast ReRoute*

The main focus was Loop Free Alternates (LFA) which is the only standardized and readily available IPFRR technology. At least two major router vendors are offering LFA-based IPFRR support out of the box [108, 109].

Claim 3.1. *The LFA coverage for $|V| \geq 2$ is at most*

$$\eta(G) \leq \frac{|V|}{|V|-1}(\Delta - 2) + \frac{2}{|V|-1} \ ,$$

where Δ denotes the average nodal degree, and at least

$$\eta(G) \geq \frac{|V|}{|V|-1}\frac{\frac{\Delta}{2}-1}{\Delta_{\max}-1} + \frac{1}{(|V|-1)(\Delta_{\max}-1)} \ .$$

where $\Delta_{\max}$ denotes the maximal nodal degree [14, 117, Theorems 17 and 18].

The bounds are for sparse networks. Next, we ask how to add edges to a graph to achieve full LFA coverage.

Claim 3.2. *The LFA graph extension problem for both the uniform cost and weighted graph cases are NP-complete [12] (See also the formal problem definitions at Definition 4 and 7, and for the proof Theorems 19 and 20).*

The proofs reduce these problems to minimal set cover through reversible transformations. Thus the heuristic approaches developed for minimal set cover can solve the LFA graph extension problem as well. Moreover, it is an efficient approach in the practice [12, 130].

We also focus on Protection Routing (PR), which requires a centralized unit that controls the forwarding table at each router but it still can be implemented with OpenFlow routers. Kwong, et. al [115] investigated the relationship between graph connectivity and 100% network failure protection Protection Routing. The next thesis is related to their Conjecture 4.1 [115]. We call two spanning trees *completely independent* if the paths in the two trees between any node pair is node-disjoint, apart from their end nodes [121] (see also Definition 12).

Claim 3.3. *A graph with two completely independent spanning trees is protectable with Protection Routing [131, Theorem 22].*

Bibliography

[1] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, and M. Jaeger, "General availability model for multilayer transport networks," in *Proc. IEEE DRCN*, Lacco Ameno, Italy, Oct. 16-19, 2005.

[2] P.-H. Ho, J. Tapolcai, and H. T. Mouftah, "On optimal diverse routing for shared protection in mesh WDM networks," *IEEE Trans. Reliability*, vol. 53, no. 6, pp. 2216–225, Jun. 2004.

[3] P.-H. Ho, J. Tapolcai, and T. Cinkler, "Segment shared protection in mesh communication networks with bandwidth guaranteed tunnels," *IEEE/ACM Trans. Networking*, vol. 12, no. 6, pp. 1105–1118, December 2004.

[4] J. Tapolcai, P. Chołda, T. Cinkler, K. Wajda, A. Jajszczyk, and D. Verchere, "Quantification of resilience and quality of service," in *Proc. IEEE International Conference on Communications (ICC)*, Istanbul, Turkey, Jun. 2006, pp. 477–482.

[5] J. Tapolcai, B. Wu, and P.-H. Ho, "On monitoring and failure localization in mesh all-optical networks," in *Proc. IEEE INFOCOM*, Rio de Janero, Brasil, 2009, pp. 1008–1016.

[6] J. Tapolcai, L. Rónyai, and P.-H. Ho, "Optimal solutions for single fault localization in two dimensional lattice networks," in *IEEE INFOCOM Mini-Symposium*, San Diego, CA, USA, 2010.

[7] J. Tapolcai, P.-H. Ho, P. Babarczi, and L. Rónyai, "On achieving All-Optical failure restoration via monitoring trails," in *Proc. IEEE INFOCOM Mini-Symposium*, Turin, Italy, USA, Apr. 2013.

[8] J. Tapolcai, B. Wu, P.-H. Ho, and L. Rónyai, "A novel approach for failure localization in all-optical mesh networks," *IEEE/ACM Trans. Networking*, vol. 19, no. 1, pp. 275 –285, feb 2011.

[9] P. Babarczi, J. Tapolcai, and P.-H. Ho, "Adjacent link failure localization with monitoring trails in all-optical mesh networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 907 – 920, June 2011.

[10] J. Tapolcai, P.-H. Ho, L. Rónyai, and B. Wu, "Network-wide local unambiguous failure localization (NWL-UFL) via monitoring trails," *IEEE/ACM Transactions on Networking*, 2012.

[11] J. Tapolcai, P.-H. Ho, L. Rónyai, P. Babarczi, and B. Wu, "Failure localization for shared risk link groups in all-optical mesh networks using monitoring trails," *IEEE/OSA J. Lightwave Technol.*, vol. 29, no. 10, pp. 1597 –1606, may 2011.

[12] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, "IP fast ReRoute: loop free alternates revisited," in *Proc. IEEE INFOCOM*, Shanghai, P.R. China, 4 2011.

[13] J. Tapolcai and G. Rétvári, "Router virtualization for improving IP-level resilience," in *Proc. IEEE INFOCOM*, Turin, Italy, USA, Apr. 2013.

[14] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, and A. Császár, "Optimizing IGP link costs for improving IP-level resilience," in *Proc. International Workshop on Design Of Reliable Communication Networks (DRCN)*, Krakow, Poland, 2011, pp. 62–69.

[15] A. Atlas, R. Kebler, M. Konstantynowicz, G. Enyedi, A. Csaszar, R. White, and M. Shand, *An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees*, Internet-Draft, Standards Track status Std., 2011.

[16] Csaszar, G. Enyedi, J. Tantsura, S. Kini, J. Sucec, and S. Das, *IP Fast Re-Route with Fast Notification*, Internet-Draft, Standards Track status Std., 2011.

[17] C. Mas, I. Tomkos, and O. Tonguz, "Failure location algorithm for transparent optical networks," *IEEE J. Select. Areas Commun.*, vol. 23, no. 8, pp. 1508–1519, 2005.

[18] H. Zeng and C. Huang, "Fault detection and path performance monitoring in meshed all-optical networks," in *Proc. IEEE GLOBECOM*, vol. 3, 2004, pp. 2014–2018.

[19] H. Zeng, C. Huang, and A. Vukovic, "A Novel Fault Detection and Localization Scheme for Mesh All-optical Networks Based on Monitoring-cycles," *Photonic Network Communications*, vol. 11, no. 3, pp. 277–286, 2006.

[20] H. Zeng and A. Vukovic, "The variant cycle-cover problem in fault detection and localization for mesh all-optical networks," *Photonic Network Communications*, vol. 14, no. 2, pp. 111–122, 2007.

[21] B. Wu, K. Yeung, B. Hu, and P. H. Ho, "M^2-CYCLE: an Optical Layer Algorithm for Fast Link Failure Detection in All-Optical Mesh Networks," *Elservier Computer Networks*, vol. 55, no. 3, pp. 748 – 758, 2011.

[22] B. Wu, K. Yeung, and P.-H. Ho, "Monitoring cycle design for fast link failure localization in all-optical networks," *IEEE/OSA J. Lightwave Technol.*, vol. 27, no. 10, pp. 1392–1401, 2009.

[23] C. Li, R. Ramaswami, I. Center, and Y. Heights, "Automatic fault detection, isolation, and recovery in transparentall-optical networks," *IEEE/OSA J. Lightwave Technol.*, vol. 15, no. 10, pp. 1784–1793, 1997.

[24] S. Stanic, S. Subramaniam, H. Choi, G. Sahin, and H. Choi, "On monitoring transparent optical networks," in *Int. Conference on Parallel Processing Workshops (ICPPW '02)*, 2002, pp. 217–223.

[25] Y. Wen, V. Chan, and L. Zheng, "Efficient fault-diagnosis algorithms for all-optical WDM networks with probabilistic link failures," *IEEE/OSA J. Lightwave Technol.*, vol. 23, pp. 3358–3371, 2005.

[26] C. Assi, Y. Ye, A. Shami, S. Dixit, and M. Ali, "A hybrid distributed fault-management protocol for combating single-fiber failures in mesh based DWDM optical networks," in *Proc. IEEE GLOBECOM*, 2002, pp. 2676–2680.

[27] B. Wu, P.-H. Ho, and K. Yeung, "Monitoring trail: On fast link failure localization in all-optical WDM mesh networks," *IEEE/OSA J. Lightwave Technol.*, vol. 27, no. 18, pp. 4175–4185, 2009.

[28] M. Maeda, "Management and control of transparent optical networks," *IEEE J. Select. Areas Commun.*, vol. 16, no. 7, pp. 1008–1023, 1998.

[29] P. Demeester, M. Gryseels, A. Autenrieth, C. Brianza, L. Castagna, G. Signorelli *et al.*, "Resilience in multilayer networks," *IEEE Commun. Mag.*, vol. 37, no. 8, pp. 70–76, 1999.

[30] N. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V. Chan, "Non-Adaptive Fault Diagnosis for All-Optical Networks via Combinatorial Group Testing on Graphs," in *Proc. IEEE INFOCOM*, 2007, pp. 697–705.

[31] B. Wu, P.-H. Ho, K. Yeung, J. Tapolcai, and H. Mouftah, "Optical Layer Monitoring Schemes for Fast Link Failure Localization in All-Optical Networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 1, pp. 114 –125, quarter 2011.

[32] C. Mas and P. Thiran, "An efficient algorithm for locating soft and hard failures in WDM networks," *IEEE J. Select. Areas Commun.*, vol. 18, no. 10, pp. 1900–1911, 2002.

[33] J. Maisonneuve, "Nonstop routing in highly available networks," in *Proc. IEEE DRCN*, Banff, Canada, Oct. 2003, pp. 228–235.

[34] A. Haddad, E. Doumith, and M. Gagnaire, "A fast and accurate meta-heuristic for failure localization based on the monitoring trail concept," *Telecommunication Systems*, pp. 1–12, 2011.

[35] S. Ahuja, S. Ramasubramanian, and M. Krunz, "Single link failure detection in all-optical networks using monitoring cycles and paths," *IEEE/ACM Trans. Networking*, vol. 17, no. 4, pp. 1080–1093, 2009.

[36] ——, "SRLG failure localization in optical networks," *IEEE/ACM Trans. Networking*, vol. 19, no. 4, pp. 989–999, 2011.

[37] E. A. Doumith, S. A. Zahr, and M. Gagnaire, "Monitoring-tree: An innovative technique for failure localization in WDM translucent networks," in *Proc. IEEE GLOBECOM*, 2010, pp. 1–6.

[38] B. Wu, P.-H. Ho, J. Tapolcai, and P. Babarczi, "Optimal allocation of monitoring trails for fast SRLG failure localization in all-optical networks," in *Proc. IEEE GLOBECOM*, 2010.

[39] E. Moghaddam, J. Tapolcai, D. Mazroa, and . Hosszu, "Physical impairment of monitoring trails in all optical transparent networks," in *Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2011.

[40] B. Wu, P.-H. Ho, J. Tapolcai, and X. Jiang, "A novel framework of fast and unambiguous link failure localization via monitoring trails," in *IEEE INFOCOM WIP*, San Diego, 2010.

[41] P. Babarczi, J. Tapolcai, and P.-H. Ho, "SRLG failure localization with monitoring trails in all-optical mesh networks," in *Proc. International Workshop on Design Of Reliable Communication Networks (DRCN)*, Krakow, Poland, 2011, pp. 188–195.

[42] S. Stanic, S. Subramaniam, G. Sahin, H. Choi, and H. A. Choi, "Active monitoring and alarm management for fault localization in transparent all-optical networks," *IEEE Trans. on Network and Service Management*, vol. 7, no. 2, pp. 118–131, 2010.

[43] C. Machuca and M. Kiese, "Optimal placement of monitoring equipment in transparent optical networks," in *Proc. IEEE DRCN*, 2009, pp. 1–6.

[44] R. Diestel, *Graph theory*. Springer New York, 2000.

[45] H. N. Gabow and H. H. Westermann, "Forests, frames, and games: Algorithms for matroid sums and applications," *Algorithmica*, vol. 7, no. 1, pp. 465–497, 1992.

[46] E. Lucas, *Recreations Mathematiques*. Gauthier-Villars, Paris, 1893.

[47] B. Alspach, "The wonderful walecki construction," *Bull. Inst. Combin. Appl*, vol. 52, pp. 7–20, 2008.

[48] W. Tutte, "On the problem of decomposing a graph into n connected factors," *Journal of the London Mathematical Society*, vol. 1, no. 1, pp. 221–230, 1961.

[49] C. Nash-Williams, "Edge-disjoint spanning trees of finite graphs," *Journal of the London Mathematical Society*, vol. 1, no. 1, pp. 445–450, 1961.

[50] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge University Press, 1994.

[51] Y. Zhao, S. Xu, X. Wang, and S. Wang, "A new heuristic for monitoring trail allocation in all-optical WDM networks," in *Proc. IEEE GLOBECOM*, dec. 2010, pp. 1 –5.

[52] Y. Zhao, S. Xu, B. Wu, X. Wang, and S. Wang, "Monitoring trail allocation in all-optical networks with the random next hop policy," in *IEEE High Performance Switching and Routing (HPSR)*, 2012, pp. 192–197.

[53] J. Tapolcai, "Web page on m-trail/tree design: simulation environments, examples and technical reports," http://opti.tmit.bme.hu/∼tapolcai/mtrail.

[54] F. K. Hwang and V. T. Sós, "Non-adaptive hypergeometric group testing," *Studia Sci. Math. Hungar*, vol. 22, pp. 257–263, 1987.

[55] D. Du and F. K. Hwang, *Combinatorial Group Testing and Its Applications*. World Scientific, 2000.

[56] D. Eppstein, M. Goodrich, and D. Hirschberg, "Improved combinatorial group testing for real-world problem sizes," in *Workshop on Algorithms And Data Structures (WADS)*. Waterloo, ON, Canada: Springer, Aug. 2005, pp. 86–98.

[57] J. W. Suurballe, "Disjoint Paths in a Network," *Networks*, vol. 4, pp. 125–145, 1974.

[58] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

[59] "LEMON: A C++ Library for Efficient Modeling and Optimization in Networks," http://lemon.cs.elte.hu.

[60] H. Zeng, C. Huang, and A. Vukovic, "A novel fault detection and localization scheme for mesh all-optical networks based on monitoring-cycles," *Photonic Communication Networking*, pp. 277–286, 2006.

[61] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, no. 6, pp. 16–23, 2000.

[62] K. Lee and E. Modiano, "Cross-layer survivability in wdm-based networks," in *Proc. IEEE INFOCOM*, 2009, pp. 1017–1025.

[63] M. Médard, R. Barry, S. Finn, W. He, and S. Lumetta, "Generalized loop-back recovery in optical mesh networks," *IEEE/ACM Trans. Networking*, vol. 10, no. 1, p. 164, 2002.

[64] N. Ogino and H. Nakamura, "All-optical monitoring path computation based on lower bounds of required number of paths," in *IEEE ICC*, 2011.

[65] M. Mao and K. Yeung, "Super Monitor Design for Fast Link Failure Localization in All-Optical Networks," in *IEEE ICC*, 2010.

[66] D. Aldous, "The random walk construction of uniform spanning trees and uniform labelled trees," *SIAM Journal on Discrete Mathematics*, vol. 3, no. 4, pp. 450–465, 1990.

[67] A. Broder, "Generating random spanning trees," in *Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1989, pp. 442–447.

[68] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proc. Int. Network Optimization Conference (INOC)*, April 2007.

[69] D. Papadimitriou and E. Mannie, "Analysis of generalized multi-protocol label switching GMPLS-based recovery mechanisms (including protection and restoration)," RFC 4428, March, Tech. Rep., 2006.

[70] S. S. Lumetta, M. Médard, and Y.-C. Tseng, "Capacity versus robustness: A tradeoff for link restoration in mesh networks," *IEEE/OSA J. Lightwave Technol.*, vol. 18, no. 12, pp. 1765–1775, December 2000.

[71] W. D. Grover, "The protected working capacity envelope concept: An alternate paradigm for automated service provisioning," *IEEE Commun. Mag.*, vol. 42, no. 1, pp. 62–69, january 2004.

[72] D. A. Schupke, W. D. Grover, and M. Clouqueur, "Strategies for enhanced dual failure restorability with static or reconfigurable p-cycle networks," in *Proc. IEEE ICC*, Paris, France, Jun. 2004, pp. 1628–1633.

[73] B. Wu, K. Yeung, and P.-H. Ho, "Ilp formulations for p-cycle design without candidate cycle enumeration," *IEEE/ACM Trans. Networking*, vol. 18, no. 1, pp. 284–295, 2010.

[74] S. Sebbah and B. Jaumard, "P-cycle based dual failure recovery in WDM mesh networks," in *Proc. IFIP Working Conference on Optical Network Design & Modelling (ONDM)*, 2009.

[75] R. R. Iraschko, M. MacGregor, and W. D. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh survivable networks," *IEEE/ACM Trans. Networking*, pp. 325–336, June 1998.

[76] Y. Xiong and L. Mason, "Restoration strategies and spare capacity requirements in self-healing atm networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 1, pp. 98–110, Oct. 1999.

[77] H. Choi, S. Subramaniam, and H. Choi, "Loopback recovery from double-link failures in optical mesh networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 6, pp. 1119–1130, 2004.

[78] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. Networking*, vol. 13, no. 1, pp. 198–211, Feb. 2005.

[79] S. Ramasubramanian and A. Harjani, "Comparison of failure dependent protection strategies in optical networks," *Photonic Network Communications*, vol. 12, no. 2, pp. 195–210, 2006.

[80] D. Wang and G. Li, "Efficient distributed bandwidth management for mpls fast reroute," *IEEE/ACM Trans. Networking*, vol. 16, no. 2, pp. 486–495, 2008.

[81] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to rsvp-te for lsp tunnels," IETF RFC 4090, May 2001.

[82] J. Doucette and W. D. Grover, "Comparison of mesh protection and restoration schemes and the dependency on graph connectivity," in *Proc. IEEE DRCN*, Oct. 2001, pp. 121–128.

[83] Y. Liu and D. Tipper, "Spare capacity allocation for non-linear link cost and failure-dependent path restoration," in *Proc. IEEE DRCN*, 2001.

[84] R. R. Iraschko and W. D. Grover, "A highly efficient path-restoration protocol for management of optical network transport integrity," *IEEE J. Select. Areas Commun.*, vol. 18, no. 5, pp. 779–794, May 2000.

[85] W. Grover, J. Doucette, M. Clouqueur, D. Leung, and D. Stamatelakis, "New options and insights for survivable transport networks," *IEEE Commun. Mag.*, vol. 40, no. 1, pp. 34–41, Jan. 2002.

[86] R. Martin, M. Menth, and K. Canbolat, "Capacity requirements for the one-to-one backup option in mpls fast reroute," in *Proc. BroadNets*, San Jose,CA, Oct. 2006.

[87] M. Frederick, P. Datta, and A. Somani, "Sub-graph routing: a generalized fault-tolerant strategy for link failures in wdm optical networks," *Computer Networks*, vol. 50, no. 2, pp. 181–199, 2006.

[88] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks, part ii - restoration," in *Proc. IEEE ICC*, 1999, pp. 2023–2030.

[89] H. Wang, E. Modiano, and M. Médard, "Partial path protection for wdm networks: End-to-end recovery using local failure information," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, 2002, pp. 719–725.

[90] W. D. Grover, *Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Upper Saddle River, New Jersey: Prentice Hall PTR, 2003.

[91] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. The Morgan Kaufmann Series in Networking, 2004.

[92] B. Wu, P.-H. Ho, K. Yeung, J. Tapolcai, and H. Mouftah, "CFP: Cooperative fast protection," *IEEE/OSA J. Lightwave Technol.*, vol. 28, no. 7, pp. 1102 –1113, apr. 2010.

[93] S. Maesschalck, D. Colle, I. Lievens, M. Pickavet, P. Demeester, C. Mauz, M. Jaeger, R. Inkret, B. Mikac, and J. Derkacz, "Pan-European optical transport networks: an availability-based comparison," *Photonic Network Communications*, vol. 5, no. 3, pp. 203–225, 2003.

[94] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, 2005.

[95] M. Shand and S. Bryant, "IP Fast Reroute framework," RFC 5714, Jan 2010.

[96] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-Free Alternates," RFC 5286, 2008.

[97] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing," in *INFOCOM*, 2004.

[98] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in *INFOCOM*, 2005.

[99] G. Enyedi, G. Rétvári, and T. Cinkler, "A novel loop-free IP fast reroute algorithm," in *EUNICE*, 2007.

[100] I. Hokelek, M. Fecko, P. Gurung, S. Samtani, S. Cevher, and J. Sucec, "Loop-free IP Fast Reroute using local and remote LFAPs," Internet Draft, Feb 2008.

[101] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Multiple routing configurations for fast IP network recovery," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 473–486, 2009.

[102] T. Čičic, A. F. Hansen, and O. K. Apeland, "Redundant trees for fast IP recovery," in *Broadnets*, 2007, pp. 152–159.

[103] A. Li, X. Yang, and D. Wetherall, "SafeGuard: safe forwarding during route changes," in *ACM CoNEXT*, 2009, pp. 301–312.

[104] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, "IP Fast Reroute using tunnels," Internet Draft, Nov 2007.

[105] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using Not-via addresses," Internet Draft, March 2010.

[106] G. Enyedi, P. Szilágyi, G. Rétvári, and A. Császár, "IP Fast ReRoute: lightweight Not-Via without additional addresses," in *INFOCOM Mini-conf*, 2009.

[107] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of NotVia," in *ACM CoNEXT*, 2007.

[108] Cisco Systems, "Cisco ios xr routing configuration guide, release 3.7," 2008.

[109] Juniper Networks, "Junos 9.6 routing protocols configuration guide," 2009.

[110] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," in *ACM CoNEXT*, 2005, pp. 244–245.

[111] S. Previdi, "IP fast reroute technologies," APRICOT, 2006.

[112] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *IEEE Comsware*, 2007.

[113] M. Menth, M. Hartmann, R. Martin, T. Čičić, and A. Kvalbein, "Loop-free alternates and not-via addresses: A proper combination for IP fast reroute?" *Comput. Netw.*, vol. 54, no. 8, pp. 1300–1315, 2010.

[114] C. Filsfils *et al.*, "LFA applicability in SP networks," Internet Draft, March 2010.

[115] K.-W. Kwong, L. Gao, R. Guerin, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 5, pp. 1543 –1556, oct. 2011.

[116] C. Reichert and T. Magedanz, "Topology requirements for resilient IP networks," in *MMB*, 2004, pp. 379–388.

[117] L.Csikor, G. Rétvári, and J. Tapolcai, "Optimizing igp link costs for improving ip-level resilience with loop-free alternates," *Computer Communications Journal*, 2012.

[118] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Comm. Mag.*, vol. 40, no. 10, pp. 118–124, Oct 2002.

[119] G. Swallow, S. Bryant, and L. Andersson, "Avoiding equal cost multipath treatment in MPLS networks," RFC 4928, June 2007.

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

[121] T. Hasunuma, "Completely independent spanning trees in the underlying graph of a line digraph," *Discrete Mathematics*, vol. 234, no. 1-3, pp. 149–157, 2001.

[122] ——, "Completely independent spanning trees in maximal planar graphs," in *Graph-Theoretic Concepts in Computer Science*. Springer, 2002, pp. 235–245.

[123] C. Oliveira and P. Pardalos, *Mathematical Aspects of Network Routing Optimization*. Springer Verlag, 2011, vol. 53.

[124] M. Resende and P. Pardalos, *Handbook of optimization in telecommunications*. Springer Verlag, 2006, vol. 10.

[125] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler, "Improving the resilience in IP networks," in *Workshop on High Performance Switching and Routing (HPSR)*. IEEE, 2003, pp. 91–96.

[126] C. Reichert, Y. Glickmann, and T. Magedanz, "Two routing algorithms for failure protection in ip networks," in *IEEE Symposium on Computers and Communications (ISCC)*, 2005, pp. 97–102.

[127] S. Ray, R. Guérin, K. Kwong, and R. Sofia, "Always acyclic distributed path computation," *IEEE/ACM Transactions on Networking (ToN)*, vol. 18, no. 1, pp. 307–319, 2010.

[128] Y. Ohara, S. Imahori, and R. Van Meter, "Mara: Maximum alternative routing algorithm," in *IEEE INFOCOM 2009*, 2009, pp. 298–306.

[129] J. Tapolcai, L. Rónyai, and P.-H. Ho, "Link fault localization using bi-directional m-trails in all-optical mesh networks," *IEEE Transactions on Communications*, 2012.

[130] M. Nagy, J. Tapolcai, and G. Retvari, "Optimization methods for improving IP-level fast protection for local shared risk groups with loop-free alternates," *Telecommunication Systems*, 2012.

[131] J. Tapolcai, "Sufficient conditions for protection routing in IP networks," *Springer Optimization Letters*, 2012.

Index