Multi-link Failure Localization via Monitoring Bursts

Mohammed L. Ali, Pin-Han Ho, János Tapolcai, and Suresh Subramaniam

***Abstract*—This paper introduces a novel monitoring trail (m-trail) allocation method for achieving local unambiguous failure localization under the monitoring burst (m-burst) framework, in which a single monitoring node (MN) can localize any multi-link failure with up to d links in a $(d+1)$-connected network by inspecting the optical bursts traversing through it. Specifically, the proposed m-trail allocation method ensures that any healthy link is traversed by at least one uninterrupted m-trail during a multi-link failure, which can be achieved by launching no more than $(d+1)$ link-disjoint m-trails per link originating from the MN. Based on such a sufficient condition, we study the performance of the m-burst framework by solving an integer linear program (ILP) and a novel heuristic, and implement the method for up to three link failures. To avoid high computation complexity in solving the ILP, a heuristic algorithm is developed for deriving $(d+1)$ link-disjoint m-trails between the MN and each link of the network. Numerical results show that the proposed methods yield significantly better performance than previous methods in the reference backbone network topologies.**

***Index Terms*—Disjoint path; Monitoring delay; Monitoring resource; M-trail; Multi-link; Single-link; UFL.**

I. Introduction

Fault management has long been a critical task in Internet control and management, and becomes even more challenging in an all-optical network. For this purpose, the link management protocol (LMP) was extended under the GMPLS framework [1]. But due to the employment of sequential coordination between adjacent nodes along each lightpath, it is subject to long fault management delay and high protocol complexity.

As a remedy, link-based monitoring [2–4] has been considered in the literature, where every link is exclusively monitored via a single-hop supervisory lightpath (S-LP) launched with constant optical signal. A more advanced version of such a scheme is by using a set of multi-hop S-LPs, called monitoring cycle (m-cycle or MC) [4–7], monitoring trail (m-trail) [8,9], or monitoring tree (m-tree) [10–12]. A monitoring node (MN) that terminates an S-LP can obtain the failure status of the group of links traversed by the S-LP. By properly allocating a set of S-LPs, the network controller can unambiguously localize any multi-link failure according to the collected alarms issued by the MNs.

For efficient integration with any shared protection or restoration method in a $(d+1)$-connected network topology, the typical goal is to localize every multi-link failure with up to d links. The failure localization schemes based on multi-hop S-LPs generally take a large amount of wavelength links (WLs), especially when multi-link failures are also considered. For example, [13] has indicated that it takes more than 10 WLs per link on average for the S-LPs and more than hundreds of transmitters to localize up to three link failures. Such high monitoring resource consumption is not tolerable under any circumstances.

To reduce WL consumption, Harvey *et al.* and Wen *et al.* suggested using an optical probe instead of a static optical signal to inspect each S-LP [11,14]. If there is no faulty link along the S-LP, the permissible probe will arrive at the receiver, and be lost otherwise. The failure localization decision is made by concluding all the inspection results at a remote coordinator. Although effective in reducing WL consumption, it is still subject to electronic signaling for alarm collection, which in turn causes long monitoring delay, and is therefore unsuitable for real-time fault management.

Several studies suggest monitoring a set of S-LPs terminated at a common MN in order to completely remove the alarm dissemination/collection complexity [2,5,15,16]. The method in [5] ensures that for each pair of failure states, there exists at least one S-LP that is interrupted by only one of the failure states. Moreover, each failure must be traversed[1] by at least one S-LP. Thus, the number of required S-LPs is upper bounded by $|E|^{2d}$, where $|E|$ is the number of links in the network, and $|E|^d$ is the number of failure states with up to d links. The studies in [15,16] suggest that a node can obtain the on–off status of an S-LP by tapping the optical signal, by which the failure

Manuscript received April 3, 2014; revised August 11, 2014; accepted August 25, 2014; published October 6, 2014 (Doc. ID 209149).

Mohammed L. Ali and Pin-Han Ho are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

János Tapolcai (e-mail: tapolcai@tmit.bme.hu) is with the MTA-BME Future Internet Research Group, Budapest University of Technology and Economics (BME), Budapest, Hungary.

Suresh Subramaniam is with the Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052, USA.

http://dx.doi.org/10.1364/JOCN.6.000952

[1]A multi-link failure is considered to be traversed by an S-LP if at least one of the failed links is traversed by the S-LP.

1943-0620/14/110952-13$15.00/0

localization decision can be made locally at each node without taking any alarm dissemination overhead.

The framework of monitoring burst (m-burst) was introduced in [17] as a remedy to all the observed problems. With m-burst, a set of S-LPs (termed m-trails in the following context) is derived such that all of them are terminated at a single MN and each link is traversed by a unique subset of m-trails. This is referred to as the scenario of local unambiguous failure localization (L-UFL) that was defined in [2,16]. Instead of launching a persistent optical signal along each m-trail, short-duration optical bursts (one for each m-trail) are sent from the MN to inspect the failure status of the m-trail. By synchronizing the switching fabric configuration of intermediate nodes of the m-trails in advance, each network link can reserve as few as a single WL along which multiple optical bursts can be multiplexed in the time domain. Accordingly, the MN follows a specific schedule of launching the bursts for each m-trail in order to achieve collision-free probing for all the m-bursts during a monitoring period.

Failure localization latency is defined in [18] as the latency with which the MN can successfully probe all the m-trails to obtain the failure status of the network without incurring any burst collision. Note that with shorter failure localization latency, the network control plane can initiate the traffic recovery phase earlier, which is essential in achieving high robustness and service continuity. To minimize the failure localization latency, a new problem for m-trail allocation and burst scheduling is formulated in [18], in which a set of m-trails and the timing of launching the burst for each m-trail are jointly determined, in order to achieve the maximum parallelism of burst traversals. The basic idea of the method in [18] is to ensure that in case of any multi-link failure, each remaining link is traversed by at least one m-trail that is disjoint from the fault, which yields the number of m-trails as $O(|E|^d)$.

Obviously, with more m-trails, less parallelism can be achieved, causing longer failure localization latency. In this paper, we improve the L-UFL m-trail allocation method of [18] by having no more than $(d+1)|E|$ m-trails in the solution. The basic idea of the proposed approach is to ensure that each link e is traversed by $(d+1)$ m-trails that are link-disjoint everywhere except on link e, such that when any d link fails, each remaining link is traversed by at least one m-trail that is disjoint from the failure. This yields a significant reduction of the upper bound on the number of consumed m-trails from $O(|E|^d)$ to $(d+1)|E|$. Note that the reduction of m-trails not only reduces the monitoring delay under the proposed m-burst framework, but also greatly simplifies the network control and management.

We will prove a sufficient condition for m-trail allocation and provide an integer linear program (ILP) that implements the proposed method. A heuristic algorithm based on the sufficient condition will be developed as a countermeasure of the huge computation complexity due to solving the ILP. We will show that the proposed method can significantly outperform the previous methods reported in [5,18].

The rest of the paper is organized as follows: Related work is discussed in Section II. The problem formulation of the m-trail allocation and burst scheduling tasks is given in Section III. Theoretical analysis of the problem and the proposed solution is given in Section IV. Two heuristic algorithms, one for m-trail allocation and another for m-burst scheduling, are provided in Section V. Numerical experiments are conducted, and the results are given in Section VI. Section VII concludes the paper. Appendix A provides an ILP to solve the m-trail allocation problem.

II. Related Work

The failure models used for survivability in backbone networks can be classified into two categories:

1) Sparse-shared risk link group (SRLG) or regional failures, where the SRLGs are assumed to have multiple links geographically close to each other. This is modeled with a few (and often disjoint) SRLGs. The challenge is typically how to localize a large SRLG in a graph with low connectivity.
2) Multi-link failures or dense SRLG, where every possible failure with up to d links is considered. Thus, $d=1$ corresponds to single-link failures, $d=2$ corresponds to both single- and double-link failures, etc. This can be modeled with many overlapping SRLGs. Here the graph connectivity is assumed to be $d+1$, and the main challenge is how to localize many overlapping SRLGs.

The two scenarios require very different methods to solve. In this study we focus on the second case. Henceforth in the paper, the terms multi-link failure and SRLG failure are used interchangeably.

Reference [11] studied nonadaptive probing schemes based on combinatorial group testing (CGT). The methods find fault-free link sets in a network G and identify probes to localize faulty links using each link set as a hub. The most efficient method is for the network topologies that can accommodate at least $(d+1)$ edge-disjoint spanning trees. The minimum number of probes $L^*(G,d)$ to localize up to d link failures is upper bounded by $O(d^3 \log_2|E|)$. The schemes are for highly connected networks and cannot be applied in networks with limited connectivity to achieve the bounds on the minimum number of probes $L^*(G,d)$.

Reference [14] introduced an adaptive run-length probing scheme in which maximum probing length K is derived to get an approximately equal probability of a single fault and no fault in a fiber segment along a Eulerian trail. The method needs $d\log_2 K + (|E|-d+1)/K + (d-2)$ probes to find d link failures.

Reference [19] introduced an ILP for allocating m-trails optimally for unambiguous multi-link SRLG failure localization wherein decimal alarm codes for each pair of the given SRLGs are kept dissimilar. Thus the number of constraints is $O(|\Psi|^2)$, which is reduced to $O(d^2|E||\Psi|)$, where d

is the maximum number of links in any SRLG and Ψ is the set of SRLGs. The idea is further explored on a bi-directional monitoring trail (bm-trail) in [13]. In the heuristic for bm-trail allocation, each SRLG with up to d arbitrary links is first assigned a unique code from the generated nonadaptive $\bar{d}$-separable CGT codes. Then, code pairs are swapped keeping code uniqueness of the SRLGs using the greedy code swapping (GCS) method to form bm-trails in each bit position of the SRLG codes in such a way that the overall failure localization cost is minimized. The worst-case complexity of the method is in $O(d^4|V||E|^2\log^2|E|(1+h_\Delta))$, where h_Δ is the difference between the maximum and minimum Hamming weights of the generated CGT codes.

Reference [20] introduced an adjacent-link failure localization (AFL) algorithm that deals with sparse SRLGs—those that include all single-link and some adjacent-link SRLGs. The dependencies among SRLG codes are removed by graph partitioning. Then, random code assignment (RCA) and random code swapping (RCS) methods are applied to each partition to find codes for single-link failures. The unique codes of all SRLGs are derived by an OR operation of the single-link codes of all the partitions. However, [20] allows an arbitrary node to terminate an m-trail, which does not account for the scenario of L-UFL.

Reference [5] is the first study on multi-link SRLG failure localization with a single MN using MC and monitoring path (MP). The complexity of the method is upper bounded by $O(|\Psi|^2|V|\log_2|V|)$, while the required number of m-trails is bounded by $O(|\Psi|^2)$. Due to the high complexity and large number of required m-trails, it leaves room for improvement.

Reference [16] is the first study on L-UFL, and it formulated an ILP for m-trail allocation by assuming multiple independent MNs and lambda monitoring capability at an intermediate node of an m-trail. The same scenario was explored in [21], which introduced a heuristic approach for the same purpose. Reference [15] took L-UFL a step further and introduced a new scenario called network-wide L-UFL (NL-UFL), which takes every node as an MN that can perform L-UFL by inspecting the on–off status of the traversing m-trails. However, that was only for single-link failures. It is clear that the results of the above three studies, although related, cannot be used in the m-burst framework of interest in this paper.

Reference [18] introduced an m-trail allocation method for multi-link SRLG fault localization. The method ensures that during any fault event each healthy link is traversed by at least one uninterrupted m-trail, which leads the required number of m-trails to be upper bounded by $O(|\Psi|)$ or $O(|E|^d)$. Although this is an enjoyable reduction of the number of required m-trails in [5], there is room to improve further. We will introduce in this paper a new scheme of m-trail allocation with an asymptotic bound of $O(d|E|)$, which is in the same order of link-based monitoring (i.e., every link is monitored by a dedicated and single-hop m-trail), aiming to effectively minimize the fault localization latency and control/management complexity.

III. Problem Description

A. Overview of the M-Burst Framework

The m-burst framework was first reported in [17]. With a given MN, the first task is to determine a set of m-trails, denoted by $\mathfrak{M}$, starting and terminating at the MN, such that any failure of an SRLG can be determined according to the aggregated information on whether the bursts launched along each m-trail return to the MN in due course of time. For this, a single WL in each unidirectional link is allocated in the network just for failure monitoring and could be traversed by one or multiple allocated m-trails. Each m-trail is probed once in each monitoring period, and the monitoring results of all the m-trails in a monitoring period are used to determine the network failure state. In the following context, we assume the MN continues monitoring the network, resulting in the fact that the length of a monitoring period is the same as the failure localization latency. The second task is to provide a schedule for launching the m-bursts to avoid any collision between the bursts.

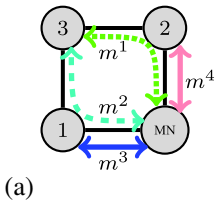
Obviously, the length of each monitoring period is determined by the set of m-trails and the timing of launching the optical bursts along each m-trail in the set. In order to minimize failure localization latency T, a joint design problem for m-trail allocation and m-burst scheduling is defined in [18]. With the set of m-trails and the burst launching time of each m-trail, the instant that the burst arrives at each intermediate node along each m-trail can be easily derived. Now, $\forall\ u \in V$ and $\forall\ m^j \in \mathfrak{M}$, and the burst arrival time to node $u \in m^j$ is $s^j + p^j_{uv}$, where s^j is the burst launching time from the MN and p^j_{uv} is the burst propagation delay from the MN to u along m-trail m^j.

Once a collision-free schedule that defines the arrival and departure timing of the bursts at each node is determined, this schedule is repeatedly executed. Accordingly at the configuration stage, node u is informed of the burst arrival time at u along each m-trail m^j traversing u, and the node can configure its switching fabric for the burst traversals at the due time.

The following subsections define the problems of m-trail allocation and m-burst scheduling, respectively.

B. M-Trail Allocation Problem

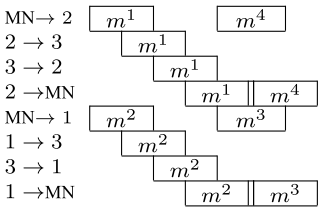
The m-trail allocation problem is to find a sequence of directed links $(v_0,v_1),(v_1,v_2),\ldots,(v_{n-2},v_{n-1}),(v_{n-1},v_n)$ for each m-trail in the network $G=(V,E)$, where $v_0=v_n$ is the MN, and $(v_i,v_{i+1})\in E$ for $i=0,\ldots,n-1$. For simplicity, if an m-trail does not form a cycle, we assume that the m-trail traverses the path in both directions (by looping back at the terminal node other than MN). This allows us to monitor links connected to degree two nodes of networks as well. However, if an m-trail forms a cycle, called an m-cycle, it has a direction in which the links are traversed. Note that an m-trail or an m-cycle is a nonsimple path that can visit a node multiple times. See also Fig. 1 for illustration.



(a)

SRLG	m^4	m^3	m^2	m^1
1,MN	0	1	1	0
MN, 2	1	0	0	1
2, 3	0	0	0	1
3, 1	0	0	1	0

(b)



(c)

Fig. 1. M-burst solution for single-link failure. (a) Topology and m-trails, (b) alarm code table (ACT), and (c) m-burst launching time.

An m-trail solution $\mathfrak{M}$ is a set of J m-trails $m^0, m^1, \ldots, m^{(J-1)}$. The on–off status of an m-trail corresponds to a specific bit position in the alarm code denoted as $[a^{(J-1)}, \ldots, a^1, a^0]$, where a^j is 1 if m^j is affected by the failed links in an SRLG, and 0 otherwise. By properly designing a set of m-trails with respect to the multi-link failures under consideration, the on–off status of the m-trails during a failure event defines an alarm code that should uniquely map the failure event to a particular SRLG. An alarm code table (ACT) maintains the mapping between alarm codes and SRLG failure events.

In the following, we define that a failure of an SRLG occurs *iff* all the links contained in the SRLG fail; on the other hand, we consider an SRLG to be traversed by an m-trail *iff* at least one link contained in the SRLG is traversed by the m-trail.

C. M-Burst Launch Time Scheduling

The m-burst scheduling problem is to find m-burst launching time s^j along each m-trail $m^j \in \mathfrak{M}$ during a monitoring period so as to minimize the fault localization latency T while avoiding burst collision altogether in the network. Let p^j_{uv} be the burst propagation delay from the MN to node u incurred by the burst along m-trail m^j until it is about to traverse unidirectional link (u,v). Thus, the burst along m-trail m^j reaches node u at $(s^j + p^j_{uv})$.

Now, $\forall\ (u,v) \in E$ and $\forall\ m^k, m^l \in \mathfrak{M}$: m^k and m^l traverse directional link (u,v), and the burst scheduler at the MN has to ensure that $|(s^k + p^k_{uv}) - (s^l + p^l_{uv})| \geq L$, where L is the burst length, to avoid burst collision in the directional link (u,v) including its end nodes u and v. Again, $T \geq T^j = s^j + tp^j + L$, $\forall\ m^j \in \mathfrak{M}$, where tp^j is the end-to-end propagation delay along m-trail m^j. Clearly, with a set of m-trails determined *a priori*, the burst scheduler can adjust only s^j to minimize T.

On the other hand, a joint design of the m-trails and burst scheduling can be performed to further reduce T at the expense of larger computation complexity.

IV. Problem Analysis

This section analyzes the m-trail allocation problem for L-UFL with a single MN and provides a theorem that serves as a basis for the subsequent ILP formulation and heuristic design.

Let the set of SRLGs and the set of m-trails traversing through SRLG ψ_j be denoted as Ψ and φ_{ψ_j}, respectively. Each SRLG is a unique set of links, and has at most d links. A necessary and sufficient condition for an eligible m-trail solution is that each SRLG is traversed by a unique subset of the m-trails in the solution. Such a condition, nonetheless, is subject to extremely high computation complexity due to its CGT process in nature. Note that there has not been any systematic approach that can efficiently solve a CGT problem even with an ILP, and this is true for our considered problem that is a special case of CGT. Thus, the previous studies can either investigate very simplified versions of the problem or take just a sufficient condition for the solution approach as a relaxation of computation complexity. For example, [16] provided an ILP for L-UFL with only single-link failures and can only be solved in very small networks, while [18] formulated an ILP that exploited a sufficient condition that the m-trails not disrupted by the faulty SRLG should traverse all the healthy links of the network; thus the ILP ensures that each link should be traversed by at least one m-trail that is disjoint from the SRLG.

Recall that the network is assumed to be $(d+1)$-connected; thus, each node u and the MN can be connected with $(d+1)$ link-disjoint paths. In the following paragraphs we will provide a theorem that can significantly improve the sufficient condition exercised in [18] with fewer m-trails, specifically less than or equal to $(d+1)|E|$ m-trails.

Theorem 1. *With every link $(u,v) \in E$ traversed by at least $(d+1)$ otherwise link-disjoint m-trails that are connected to the MN, every multi-link SRLG failure involving up to d links can be unambiguously localized at the MN by detecting disrupted m-trails only.*

Proof: In the proof we need to show that $\forall\ \psi_j, \psi_k \in \Psi$: if $\psi_j \neq \psi_k$, we have $\varphi_{\psi_j} \neq \varphi_{\psi_k}$, where $|\psi_j| \leq d$ and $|\psi_k| \leq d$. Without losing generality, assume that $|\psi_j| \leq |\psi_k|$. Thus, ψ_j may or may not be a subset of ψ_k. But ψ_k cannot be a subset of ψ_j. Therefore, we have to deal with two cases.

Case 1, $\psi_j \subset \psi_k$: This implies that all the m-trails that traverse ψ_j will also traverse ψ_k, i.e., $\varphi_{\psi_j} \subseteq \varphi_{\psi_k}$. On the other hand, in this case, $|\psi_j|$ must be less than $|\psi_k|$. Thus, $\exists (w,x) \in E : (w,x) \notin \psi_j \wedge (w,x) \in \psi_k$. As each link of the network including link (w,x) is traversed by at least $(d+1)$ otherwise link-disjoint m-trails, the SRLG ψ_k will be traversed by at least two m-trails that are link-disjoint from the SRLG ψ_j. Therefore, we have $|\varphi_{\psi_j}| < |\varphi_{\psi_k}|$.

Case 2, $\psi_j \not\subset \psi_k$: In this case $|\psi_j|$ is less than or equal to $|\psi_k|$. In either situation $\exists (u,v), (w,x) \in E : (u,v) \in \psi_j \wedge$

$(u,v) \notin \psi_k$ and $(w,x) \notin \psi_j \wedge (w,x) \in \psi_k$. As each link of the network including link (u,v) is traversed by $(d+1)$ otherwise link-disjoint m-trails, the SRLG ψ_j will be traversed by at least one m-trail that is link-disjoint from the SRLG ψ_k. Thus, we have $\varphi_{\psi_j} \nsubseteq \varphi_{\psi_k}$. Similarly, as each link of the network including link (w,x) is traversed by $(d+1)$ otherwise link-disjoint m-trails, the SRLG ψ_k will be traversed by at least one m-trail that is link-disjoint from the SRLG ψ_j. Thus, we have $\varphi_{\psi_k} \nsubseteq \varphi_{\psi_j}$.

Thus, each SRLG is traversed by a unique set of m-trails. Therefore, SRLG failure involving up to d links can be unambiguously localized at the MN by detecting disrupted m-trails only. ∎

Based on Theorem 1, an ILP for m-trail allocation is formulated in Appendix A. In the ILP-based approach for fault localization, the redundant m-trails in each ILP solution are removed by using the post-processing technique described in Subsection V.B. The scheduling algorithm will take the resultant set of m-trails as an input.

We observe that in each network involved in the numerical experiments conducted in [22], the fault localization latency derived by the heuristic burst scheduling scheme given in Algorithm 2 is very close to that derived by the ILP burst scheduling scheme proposed in [22] in solving the scheduling problem. We will therefore use the heuristic algorithm (Algorithm 2) presented in Subsection V.C for burst launch time scheduling in this paper.

V. Heuristic Algorithms

In this section, a heuristic algorithm for m-trail allocation is given in Subsection V.A. The heuristic, similarly to the ILP, is based on Theorem 1. First, it iteratively finds $(d+1)$ link-disjoint m-trails between each link and the MN of a network with a greedy method. Next, the redundant m-trails in the solution are removed as described in Subsection V.B. The resultant set of m-trails is then used as the input of the burst scheduling heuristic algorithm described in Subsection V.C.

A. M-Trail Allocation for SRLG Fault Localization

The pseudo-code of the proposed heuristic algorithm for the m-trail allocation problem is given in Algorithm 1. The basic idea is to reuse each m-trail as much as possible such that every link is traversed by at least $(d+1)$ otherwise link-disjoint m-trails, and there exists a set of m-trails that is disjoint from each SRLG while covering the rest of the network.

In line 1, the solution set of m-trails $\mathfrak{M}$ is initialized as empty. In lines 2 and 3, each undirected link of the network is assigned a distance that is the smaller one of the two shortest distances of the end nodes of the link from the MN. In line 4, links are sorted in descending order of their distances. Thus, m-trails traversing links that are far away from the MN will be derived before m-trails traversing links near the MN.

The outer *for* loop will consider each undirected link (u,v) in turn. Each iteration of the loop covers lines 5–11. In line 5, the set of m-trails $\mathfrak{M}_{uv}$ currently traversing the link (u,v) is retrieved from $\mathfrak{M}$. The intention is to reduce the total number of m-trails in the solution by reusing the largest possible set of m-trails already traversing link (u,v) but link-disjoint in all other links.

In line 6, each set of disjoint m-trails in $\mathfrak{M}_{uv}$ is found using a brute force search. During the brute force search all the possible subsets with $2,\ldots,d+1$ m-trails from $\mathfrak{M}_{uv}$ are verified whether they are link-disjoint or not in all links except link (u,v). All disjoint sets of m-trails are stored in set $\mathfrak{M}_{uv}^{ds}$.

If every set of link-disjoint m-trails in $\mathfrak{M}_{uv}^{ds}$ is smaller than $(d+1)$, it indicates that (u,v) is not yet traversed by $(d+1)$ otherwise link-disjoint m-trails. In this case, each m-trail set $\mathfrak{M}_{uv}^{j}$ in $\mathfrak{M}_{uv}^{ds}$ will be considered in turn. We tentatively assume that the m-trails in $\mathfrak{M}_{uv}^{j}$ are already found for (u,v), and we search for the remaining $d+1-|\mathfrak{M}_{uv}^{j}|$ otherwise link-disjoint m-trails in the graph $G\backslash\{(u,v) \cup \mathfrak{M}_{uv}^{j}\}$ using a minimum cost flow algorithm, e.g., Suurballe's algorithm [23], in line 7. If no set in $\mathfrak{M}_{uv}^{ds}$ can be used to find a reduced number of new link-disjoint m-trails such that link (u,v) is traversed by $(d+1)$ otherwise link-disjoint m-trails, we derive $(d+1)$ link-disjoint paths without using any existing m-trail in $\mathfrak{M}_{uv}$ in line 10. Since the network is assumed to be $(d+1)$-connected, there exist $(d+1)$ disjoint paths between the MN and each link of the network as per Menger's theorem [24]. An m-trail is formed using each new path and link (u,v), and the m-trail is added to $\mathfrak{M}$.

Algorithm 1 M-Trail Allocation for Fault Localization

Input: $G(V,E)$, MN, and d
Output: $\mathfrak{M}$
begin
1 Initialize $\mathfrak{M} \leftarrow \varnothing$
2 $\forall\, u \in V$, find shortest distance $d(u)$ from the MN
for each *undirected link* $(u,v) \in E$ **do**
3 Assign $\min\{d(u), d(v)\}$ as the distance of (u,v)
4 Sort E in descending order of the link distances
for each $(u,v) \in E$ *in descending order of distances* **do**
5 $\mathfrak{M}_{uv} \leftarrow$ the set of m-trails traversing (u,v) in $\mathfrak{M}$
6 $\mathfrak{M}_{uv}^{ds} \leftarrow$ all disjoint m-trail subsets of $\mathfrak{M}_{uv}$
if *all set in* $\mathfrak{M}_{uv}^{ds}$ *has less than* $(d+1)$ *m-trails* **then**
for each $\mathfrak{M}_{uv}^{j} \in \mathfrak{M}_{uv}^{ds}$ *in descending order of cardinalities* **do**
7 Find $d+1-|\mathfrak{M}_{uv}^{j}|$ link-disjoint paths between the MN and (u,v) in $G\backslash\{(u,v) \cup \mathfrak{M}_{uv}^{j}\}$
if *succeed* **then**
for each *disjoint path* p **do**
8 $\mathfrak{M} \leftarrow$ new m-trail $p \cup (u,v)$
9 **break** proceed to next link (u,v).
10 Find $d+1$ link-disjoint paths in $G\backslash(u,v)$ between the MN and (u,v).
for each disjoint path p **do**
11 $\mathfrak{M} \leftarrow$ new m-trail $p \cup (u,v)$

Let us derive the upper bound on the number of m-trails in a solution, $|\mathfrak{M}|$. The main *for* cycle of Algorithm 1 iterates $|E|$ times, where $|E|$ is the number of undirected links. At most $(d+1)$ disjoint paths are added to $\mathfrak{M}$ in each iteration of the loop. Thus, we have $|\mathfrak{M}| \leq (d+1)|E|$.

Now, let us derive the worst-case complexity of the algorithm. Lines 2 and 3 need $O(|V|\log_2|V| + |E|)$ in the worst case to find node distances from the MN using Dijkstra's algorithm and assign the minimum distances to the links. Line 4 needs $O(|E|\log|E|)$ steps to store the links in a heap. The outer *for* loop in lines 5–11 iterates $|E|$ times.

As m-trails derived for each link are link-disjoint in other links of the network, the number of m-trails traversing any link will be at most $|E|$. Thus, for each undirected link (u,v), $|\mathfrak{M}_{uv}| \leq |E|$. Now, to find all combinations of up to $(d+1)$ m-trails in $\mathfrak{M}_{uv}$, $O(|E|^{(d+1)})$ steps are required. To perform pairwise AND operations in a set of m-trails with cardinality $(d+1)$, $O((d+1)^2)$ steps are needed. Thus, line 6 requires at most $O((d+1)^2|E|^{(d+1)})$ steps to derive $\mathfrak{M}_{uv}^{ds}$.

The first inner *for* loop in lines 7–9 will iterate $|\mathfrak{M}_{uv}^{ds}|$ times, which is equivalent to $O(|E|^d)$ because the cardinality of the largest m-trail set in $\mathfrak{M}_{uv}^{ds}$ will be less than or equal to d inside the loop. Consequently, at most d disjoint paths for a link will be searched $O(|E|^d)$ times in line 7. Finding d disjoint paths from the MN to each link requires $O(d|V||E|)$ steps. Line 8 needs $O(d|E|)$ steps. Thus, the first inner *for* loop needs $O(d|V||E|^{d+1})$ steps. Line 10 needs $O((d+1)|V||E|)$ steps. Line 11 needs $O((d+1)|E|)$ steps.

Thus, the overall complexity of the algorithm will be $O(d(d+|V|)|E|^{(d+2)})$.

B. Postprocessing Module

Note that Theorem 1 serves as a sufficient condition for L-UFL, which yields some redundant m-trails against the optimal case. Therefore, results by the ILP given in Appendix A and Algorithm 1 should go through a postprocess in order to remove any redundant m-trail.

Once the set of m-trails $\mathfrak{M}$ is derived by solving the ILP or Algorithm 1, we have applied two established rules to remove any redundant m-trail from $\mathfrak{M}$: two or more alarm codes (there is one alarm code for each SRLG failure) should not become equal, and no alarm code should become 0 after the removal of the m-trail [4]. The postprocessing module needs $O(d|E|^{2d+1})$ steps to remove a redundant m-trail from $\mathfrak{M}$ as $|\mathfrak{M}| \leq (d+1)|E|$ and the number of SRLGs under consideration $|\Psi| \leq |E|^d$.

C. M-Burst Launch Time Scheduling

Once a set of m-trails $\mathfrak{M}$ that provides a unique code for each SRLG is derived by the ILP given in Appendix A or by the heuristic algorithm given in Subsection V.A, and the redundant m-trails are removed, the burst launching times from the MN are calculated. The set of starting times S of the m-bursts along the m-trails is derived by the burst scheduling algorithm given in Algorithm 2. The algorithm is based on Tabu search [25].

Briefly, Tabu search is a metaheuristic that starts with a current solution (found randomly or based on the specific problem instance), and then iterates as long as the current solution seems to be improving. A Tabu list keeps track of the recent current solutions in order to avoid the same solution as the current solution repeatedly. Each solution in the Tabu list is a Tabu solution and remains so for a predefined number of current solution updates. In each iteration of Tabu search, the best solution in the neighborhood of the current solution is found. Usually in permutation problems, each neighborhood is identified by pairwise swapping of the positions in the current solution. Each pairwise swapping of the positions provides another potential solution and is called a move. A move is associated with a move value. If the best move is an improvement on the current solution, it is accepted as the current solution. Otherwise, the best move that is not a Tabu solution is selected as the current solution [25].

In Algorithm 2, the m-trails in $\mathfrak{M}$ are stored in an ordered list *seq*. A sequence of m-trails is defined by the positions of m-trails in *seq* at a point in time. The primary objective of the burst scheduling heuristic is to find the minimum fault localization latency T avoiding burst collisions altogether. The near-optimal sequence of m-trails that provides minimum T is found by repeatedly swapping positions of m-trails pairwise in *seq* as per Tabu search rules [25].

In line 1, T is set to ∞. The outer *while* loop finds minimum T. Each iteration of the *while* loop has two sections: initialization in lines 2–7 and Tabu search in lines 8–16.

In line 2, the Tabu list, the ordered list *seq*, the new spin set sp_{new}, and the new propagation delay table P are initialized. Then, a position in *seq* for each m-trail is chosen randomly, and spin sp_{new}^j of the m-trail is also chosen randomly in lines 3 and 4. The spin will determine if the m-burst will traverse on-trail links in the given or opposite direction. Then, burst propagation delay p_{uv}^j from the MN to node u of each unidirectional link (u,v) traversed by the jth m-trail is calculated in line 5 considering the spin of the burst. Next, new fault localization latency T_{new} and a new set of burst launching times S_{new} are initialized using the helper function FindMaxDelay in line 6. Finally, the spin set sp, the set of burst launching times S, and the minimum fault localization latency T are updated if T_{new} is less than T in line 7.

The inner *while* loop is the Tabu search section. It has two subsections: Tabu move and update. The inner *while* loop will find minimum T_{new} for a particular spin set sp_{new} of the m-trails.

In the Tabu move subsection, lines 8–12, all the move values corresponding to the pairwise swapping of the current positions of the m-trails in *seq* are calculated by using the helper function FindMaxDelay. A move is either a Tabu or a non-Tabu move. The best Tabu and non-Tabu moves are found in the subsection that involves the most expensive calculations.

In the update subsection, lines 13–16, the relevant parameters seq, S_{new}, and T_{new} are updated based on the move values. If the maximum improvement of T_{new} is provided by the best Tabu move, it is used to update the relevant parameters. Otherwise, the best non-Tabu move is used to update the relevant parameters. Then, the Tabu list is updated for the next iteration of the inner *while* loop. Next, sp, S, and T are updated if T_{new} is less than T in line 16.

Algorithm 2 M-Burst Launch Time Scheduling

Input: $G(V,E)$, MN, n, and $\mathfrak{M}$
Output: sp, S, and T
begin
1 Initialize $T \leftarrow \infty$
While # *iteration of nonimproving* $T < n$ **do**
2 Initialize Tabu list, seq, sp_{new}, and P
for each $m^j \in \mathfrak{M}$ **do**
3 Randomly choose a free position k in seq and the spin sp^j_{new} of the m-trail
4 $seq[k] \leftarrow m^j$, $sp_{new} \leftarrow sp^j_{new}$
5 $P \leftarrow$ Find burst arrival time p^j_{uv}, $\forall\ (u,v) \in E$
6 S_{new}, $T_{new} \leftarrow$ FindMaxDelay(P, seq)
if $T_{new} < T$ **then**
7 $sp \leftarrow sp_{new}$, $S \leftarrow S_{new}$, $T \leftarrow T_{new}$
while # *iteration of nonimproving* $T_{new} < n$ **do**
8 Initialize T_{nt}, $T_t \leftarrow \infty$
for each ith and jth positions in seq **do**
9 $seq_{ij} \leftarrow$ swap positions of the m-trails
10 S_{ij}, $T_{ij} \leftarrow$ FindMaxDelay(P, seq_{ij})
if *it is a non-Tabu move* $\wedge\ T_{ij} < T_{nt}$ **then**
11 $seq_{nt} \leftarrow seq_{ij}$, $S_{nt} \leftarrow S_{ij}$, $T_{nt} \leftarrow T_{ij}$
if *it is a Tabu move* $\wedge\ T_{ij} < T_t$ **then**
12 $seq_t \leftarrow seq_{ij}$, $S_t \leftarrow S_{ij}$, $T_t \leftarrow T_{ij}$
if $T_t \le T_{new} \wedge T_t < T_{nt}$ **then**
13 $seq \leftarrow seq_t$, $S_{new} \leftarrow S_t$, $T_{new} \leftarrow T_t$
else
14 $seq \leftarrow seq_{nt}$, $S_{new} \leftarrow S_{nt}$, $T_{new} \leftarrow T_{nt}$
15 Update Tabu list
if $T_{new} < T$ **then**
16 $sp \leftarrow sp_{new}$, $S \leftarrow S_{new}$, $T \leftarrow T_{new}$

In Algorithm 2, seq can have $O(|\mathfrak{M}|!)$ number of sequences of m-trails, which is equivalent to $O(((d+1)|E|)!)$. The Tabu search technique helps to find a sequence of m-trails in seq that provides near-optimal T without searching all the sequences of m-trails.

The helper function FindMaxDelay calculates the starting times S_{sq} of m-bursts from the MN along the m-trails from the first position to the last position of the ordered list sq. Each burst is kept nonoverlapping in each unidirectional link with all other bursts along the m-trails in the preceding positions of sq. Thus, $\forall\ m^i$ in the preceding position of each m-trail m^j in sq, if burst along m^j arrives at the sending node u of a link (u,v) before the burst along m^i completely passes through u, the starting time of the burst s^j has to be increased to satisfy the condition $s^j + p^j_{uv} \ge s^i + p^i_{uv} + L$ in order to avoid collision in (u,v). In this case, checking with the m-trails in the preceding positions of m^j has to be restarted from the beginning. In other words, the function finds the earliest possible burst launching time s^j from the MN for the m-burst along each m-trail m^j. Here, L is the burst length, and p^i_{uv} and p^j_{uv} are the burst propagation delays along m-trails m^i and m^j from the MN to node u, respectively. Note that p^i_{uv}, p^j_{uv}, and L are constants from the scheduler's perspective.

The fault localization latency T_{sq} for the given sequence sq is the maximum value of $(s^j + tp^j + L)$ provided by any m-trail m^j, where tp^j is the end-to-end propagation delay along the m-trail. S_{sq} and T_{sq} are returned in line 10 of FindMaxDelay.

Function FindMaxDelay(P,sq)

Input: $G(V,E)$, MN, and $\mathfrak{M}$
Output: S_{sq}, T_{sq}
begin
1 $S_{sq} \leftarrow \emptyset$, $m^j \leftarrow sq[0]$, $s^j \leftarrow 0$
2 $S_{sq} \leftarrow s^j$, $T_{sq} \leftarrow s^j + tp^j + L$
for each $k \in \{1 \ldots (|sq|-1)\}$ **do**
3 $m^j \leftarrow sq[k]$, $s^j \leftarrow 0$
4 **for each** $l \in \{0 \ldots (k-1)\}$ **do**
$m^i \leftarrow sq[l]$, get s^i from S_{sq}
for each *unidirectional link* $(u,v) \in E$ **do**
if *both* m^i *and* m^j *visit* (u,v) **then**
5 Get p^i_{uv} and p^j_{uv} from P
if $(s^j + p^j_{uv} + L) > (s^i + p^i_{uv}) \wedge$ $(s^j + p^j_{uv}) < (s^i + p^i_{uv} + L)$ **then**
6 $s^j \leftarrow (s^i + p^i_{uv} + L - p^j_{uv})$
7 **GoTo** line 4
8 $S_{sq} \leftarrow s^j$
if $(s^j + tp^j + L) > T_{sq}$ **then**
9 $T_{sq} \leftarrow s^j + tp^j + L$
10 Return S_{sq}, T_{sq}

For each sequence of m-trails in sq, the helper function FindMaxDelay finds T_{sq} taking $\Omega(|E||\mathfrak{M}|^2)$ steps.

VI. Numerical Results

We solved the m-trail allocation problem in 10 networks using either the ILP given in Appendix A in ILOG CPLEX 11.1 or the proposed heuristic method given in Subsection V.A. The experiments consider all the SRLGs with up to arbitrary three links, where each SRLG with two or three links is node-disjoint from the arbitrarily chosen MN in each network.

The set of m-trails derived by solving either the ILP or the heuristic is used as an input to the post-process module, followed by the invocation of the m-burst scheduling heuristic in Algorithm 2, in order to find fault localization latency T. In the numerical experiments for burst scheduling, we assume that the burst length L is 20 ms, the burst propagation delay δ through any unidirectional link is 2 ms, and there is a single supervisory WL along each unidirectional link traversed by any m-trail.

By taking $r_1 = 0.1$ and $r_2 = 0.01$ for the ILP-based approach, we first consider a network with seven nodes

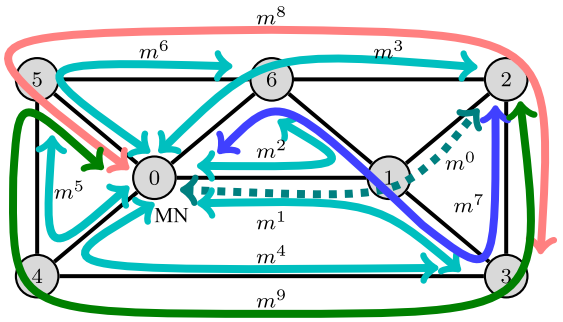


Fig. 2. M-trail set for a network with seven nodes and 12 links.

and 12 links, which has in total 96 SRLGs consisting of 12 single-link, 28 double-link, and 56 triple-link SRLGs. As shown in Fig. 2, the solution of the ILP-based approach requires 10 m-trails to form an ACT at the MN that has a unique alarm code for each SRLG. The ACT is shown partially in Table I. The alarm code of an SRLG is determined by the m-trails traversing through the SRLG. For example, link (1, 2) is traversed by m-trail m^0; thus its alarm code is 0 000 000 001, and the corresponding decimal alarm code (*Dec.*) is $2^0 = 1$. As link (3, 4) is traversed by m-trails m^4 and m^9, its alarm code is 1 000 010 000, and the corresponding *Dec.* is $2^4 + 2^9 = 528$. Similarly, link (5, 6) is traversed by m-trails m^6 and m^8; hence its alarm code is 0 101 000 000, and the corresponding *Dec.* is $2^6 + 2^8 = 320$. On the other hand, since the SRLG $\psi = \{(1,2),(3,4),(5,6)\}$ is traversed by all the m-trails mentioned above, its alarm code 1 101 010 001 is the bitwise OR of its link alarm codes, and the corresponding *Dec.* is $2^0 + 2^4 + 2^6 + 2^8 + 2^9 = 849$.

When an SRLG fails, a unique set of m-trails is disrupted, resulting in a unique alarm code. For example, if m^1, m^2, m^5, m^7, and m^9 are disrupted, the generated alarm code will be 1 010 100 110. The *Dec.* is $2^1 + 2^2 + 2^5 + 2^7 + 2^9 = 678$. Using 678 as index, the faulty SRLG can be localized as $\{(1,3),(1,6),(4,5)\}$ from the ACT given in Table I.

The starting time of the bursts s^j, total propagation delay through the m-trail tp^j, burst length L, and monitoring delay for the jth m-burst T^j in the network with seven nodes and 12 links are shown in Table II. Maximum monitoring delay, i.e., fault localization latency T, is 80 ms. Thus, T is derived in the solution from the burst along m-trail m^7.

Figure 3 illustrates the period when the m-bursts traverse through links from the MN to node 2 via nodes 6, 1, and 3, and back to the MN. Each rectangle represents the time period that the corresponding m-burst traverses

TABLE I
ALARM CODE TABLE

Dec.	SRLG	Dec.	SRLG	Dec.	SRLG
1	(1 2)	560	(3 4)(4 5)	901	(1 2)(1 6)(2 3)
7	(0 1)	561	(1 2)(3 4)(4 5)	902	(1 3)(1 6)(2 3)
48	(0 4)	658	(1 3)(3 4)	904	(2 3)(2 6)
130	(1 3)	659	(1 2)(1 3)(3 4)	905	(1 2)(2 3)(2 6)
131	(1 2)(1 3)	660	(1 6)(3 4)	906	(1 3)(2 3)(2 6)
132	(1 6)	661	(1 2)(1 6)(3 4)	908	(1 6)(2 3)(2 6)
133	(1 2)(1 6)	662	(1 3)(1 6)(3 4)	912	(2 3)(3 4)
134	(1 3)(1 6)	674	(1 3)(4 5)	913	(1 2)(2 3)(3 4)
135	(1 2)(1 3)(1 6)	675	(1 2)(1 3)(4 5)	914	(1 3)(2 3)(3 4)
136	(0 6)	676	(1 6)(4 5)	916	(1 6)(2 3)(3 4)
264	(2 6)	677	(1 2)(1 6)(4 5)	920	(2 3)(2 6)(3 4)
265	(1 2)(2 6)	678	(1 3)(1 6)(4 5)	922	(1 3)(2 6)(3 4)
320	(5 6)	690	(1 3)(3 4)(4 5)	924	(1 6)(2 6)(3 4)
321	(1 2)(5 6)	692	(1 6)(3 4)(4 5)	928	(2 3)(4 5)
328	(2 6)(5 6)	792	(2 6)(3 4)	929	(1 2)(2 3)(4 5)
329	(1 2)(2 6)(5 6)	793	(1 2)(2 6)(3 4)	930	(1 3)(2 3)(4 5)
394	(1 3)(2 6)	808	(2 6)(4 5)	932	(1 6)(2 3)(4 5)
395	(1 2)(1 3)(2 6)	809	(1 2)(2 6)(4 5)	936	(2 3)(2 6)(4 5)
396	(1 6)(2 6)	824	(2 6)(3 4)(4 5)	938	(1 3)(2 6)(4 5)
397	(1 2)(1 6)(2 6)	832	(0 5)	940	(1 6)(2 6)(4 5)
398	(1 3)(1 6)(2 6)	848	(3 4)(5 6)	944	(2 3)(3 4)(4 5)
450	(1 3)(5 6)	849	(1 2)(3 4)(5 6)	960	(2 3)(5 6)
451	(1 2)(1 3)(5 6)	856	(2 6)(3 4)(5 6)	961	(1 2)(2 3)(5 6)
452	(1 6)(5 6)	864	(4 5)(5 6)	962	(1 3)(2 3)(5 6)
453	(1 2)(1 6)(5 6)	865	(1 2)(4 5)(5 6)	964	(1 6)(2 3)(5 6)
454	(1 3)(1 6)(5 6)	872	(2 6)(4 5)(5 6)	968	(2 3)(2 6)(5 6)
458	(1 3)(2 6)(5 6)	880	(3 4)(4 5)(5 6)	976	(2 3)(3 4)(5 6)
460	(1 6)(2 6)(5 6)	896	(2 3)	978	(1 3)(3 4)(5 6)
528	(3 4)	897	(1 2)(2 3)	980	(1 6)(3 4)(5 6)
529	(1 2)(3 4)	898	(1 3)(2 3)	992	(2 3)(4 5)(5 6)
544	(4 5)	899	(1 2)(1 3)(2 3)	994	(1 3)(4 5)(5 6)
545	(1 2)(4 5)	900	(1 6)(2 3)	996	(1 6)(4 5)(5 6)

TABLE II
MONITORING DELAYS ALONG THE M-TRAILS

m^j	$s^j + tp^j + L = T^j$	m^j	$s^j + tp^j + L = T^j$
0	$40 + 8 + 20 = 68$	5	$30 + 8 + 20 = 58$
1	$0 + 8 + 20 = 28$	6	$50 + 8 + 20 = 78$
2	$20 + 8 + 20 = 48$	7	$44 + 16 + 20 = 80$
3	$0 + 8 + 20 = 28$	8	$22 + 16 + 20 = 58$
4	$50 + 8 + 20 = 78$	9	$0 + 16 + 20 = 36$

TABLE III
ILP RESULTS IN ADDITIONAL NETWORKS

Networks	No. of SRLGs	No. of M-Trails		T	
		ILP in [19]	ILP	ILP in [19]	ILP
4 nodes, 6 links	10	6	6	68	68
5 nodes, 8 links	18	9	8	68	68
6 nodes, 12 links	96	9	12	100	88

through a link. The time period is equal to propagation delay δ through the link, which is 2 ms, and burst length L, which is 20 ms, i.e., $\delta + L = 2 + 20 = 22$ ms. If one burst arrives at node u to traverse unidirectional link (u, v) and at the same moment another burst leaves node v after traversing the link, they are shown adjacent to each other. Back-to-back periods are indicated in Fig. 3 by black rectangles when two bursts traverse through a unidirectional link one immediately after the other. The m-bursts in Fig. 3 as well as all other bursts remain nonoverlapping through any link of the network. Hence, the m-bursts are collision-free throughout the network.

To verify the proposed algorithm, we run the ILP-based approach on three more networks with an arbitrarily chosen MN in each network. The simulation results in terms of the number of m-trails and the fault localization latency T are given in Table III. In each network, the method identifies a set of m-trails for unambiguous failure localization (UFL) during multi-link SRLG fault events and achieves collision-free burst scheduling by keeping the m-bursts along the m-trails nonoverlapping through any link of the network.

The simulation results by the ILP method of [19] are also included in Table III for comparison. The method derives an optimal set of m-trails that minimizes cover length keeping each SRLG alarm code unique where each m-trail has its own MN. Thus, we modified the method in order to derive the m-trail solution having a single MN. Note that the ILP in [19] does not consider the suppression on the maximum number of m-trails traversing through a single link; thus it may lead to longer latency. The method yields solutions only for networks consisting of up to six nodes and 12 links.

The proposed ILP-based approach, although better than that in [19] in terms of T, is still subject to very high computation complexity and can only be implemented in small networks. In the experiments, the ILP-based approach yields feasible solutions only for networks consisting of up to seven nodes and 12 links. As it did not return any valid solution within 24 h, we stopped running the program in even larger networks. The ILP solutions for networks with four nodes and six links, five nodes and eight links, and six nodes and 12 links are optimal, but the solution obtained within 24 h for the network with seven nodes and 12 links has an 18.65% optimality gap.

Similarly, in Table IV, the performance of the proposed heuristic method given in Subsection V.A is compared with that of the MC-1 method proposed in [5], which uses the well-known necessary and sufficient conditions of UFL based on pairwise SRLG code comparison, and the method proposed in [18] in six network topologies. The results derived by MC-1 when deployed in the m-burst framework, the method in [18], and the proposed heuristic are shown in columns H1, H2, and H3, respectively. The proposed heuristic for m-trail allocation requires significantly fewer m-trails than required by the other schemes. Moreover, the proposed heuristic method yields the best performance in terms of the shortest fault localization latency T among the methods.

The average number of m-trails traversed through each undirected link and the maximum number of m-trails traversed through any unidirectional link in six networks are compared in Figs. 4(a) and 4(b), respectively, where the proposed method outperforms the other schemes. Thus,

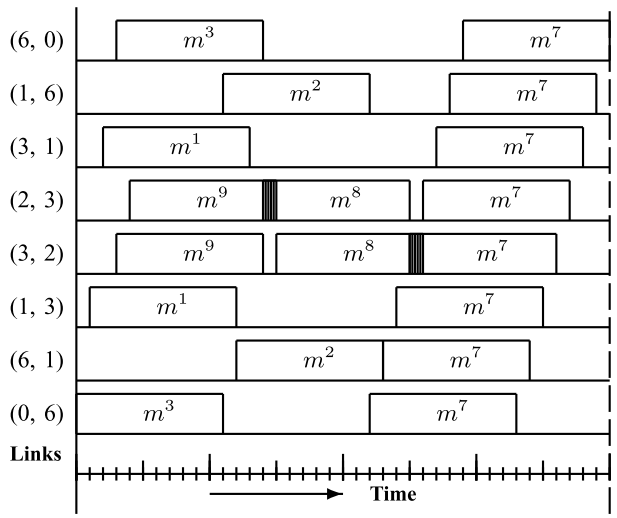


Fig. 3. Schedule of m-bursts through selected unidirectional links.

TABLE IV
PERFORMANCE OF THE HEURISTIC METHODS

Networks	No. of SRLGs	No. of M-Trails			T		
		H1	H2	H3	H1	H2	H3
8 nodes, 12 links	132	22	27	15	172	228	120
9 nodes, 14 links	179	27	24	18	154	164	114
CERNet	236	29	32	20	168	258	152
SmallNet	1162	43	64	27	336	502	216
NSFNet + 2 links	1353	68	98	37	556	876	294
Bellcore + 1 link	2053	59	86	37	346	742	220

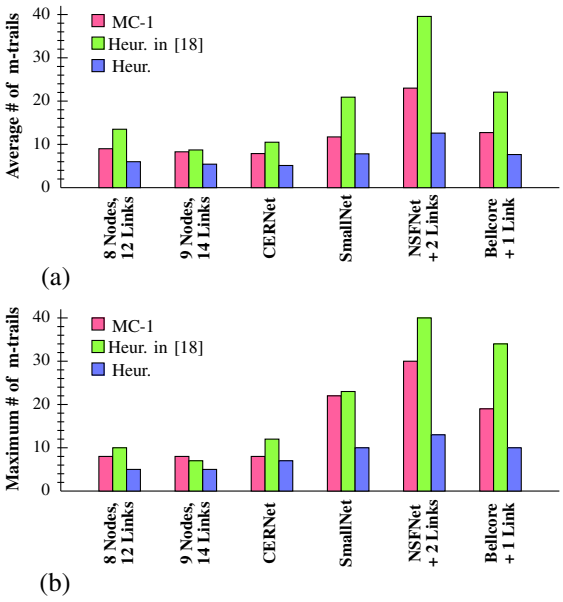


Fig. 4. M-trail traversal through network links. (a) Average number of m-trails traversed through each link. (b) The maximum number of m-trails traversed through any link.

it is not surprising that the proposed heuristic method achieves significantly lower T.

VII. Conclusions

This paper studied the m-trail allocation problem under multi-link SRLG fault localization for the m-burst framework. With the aim of minimizing the fault localization latency, the m-trail allocation problem is defined. Detailed analysis was provided, by which theoretical proof was given to show that the proposed m-trail allocation method can achieve better performance than the case by using any of the previously proposed counterparts. Experiments were conducted to verify the proposed ILP-based approach and the heuristic algorithm, and compare them with three counterparts reported in [19], [5], and [18], respectively. We conclude that the proposed heuristic algorithm can achieve significant improvement in terms of the total monitoring delay against the previously reported schemes.

Appendix A

This appendix provides an ILP for implementing the condition in Theorem 1. Different from that in [18], the developed ILP ensures that each undirected link is traversed by at least $(d+1)$ otherwise link-disjoint m-trails while each m-trail in an ILP solution visits the MN. This guarantees that each undirected link in $E\backslash\psi$, $\forall\ \psi \in \Psi$ is traversed by at least one m-trail that is link-disjoint from the SRLG ψ as $|\psi| \le d$. The derived solution is sufficient for achieving L-UFL.

The predefined constants of the ILP are δ, r_1, r_2, and J. δ is a small positive constant, which is the minimum voltage increase along an m-trail, $|E|^{-1} \ge \delta > 0$. To ensure the connectivity of the m-trails, we use the Miller–Tucker–Zemlin subtour elimination constraints [26], also called the voltage constraint technique [9]. r_1 and r_2 are two penalty parameters to mitigate the solution. J is the maximum number of allowed m-trails in an ILP solution. Moreover, i and j are indices of m-trails, where $i,j \in \{0,1,2,\ldots,J-1\}$.

The binary variables are as follows:

- m^j is 1 if the j th m-trail is in an ILP solution and 0 otherwise.
- mt^j is 1 if m^j is a trail and 0 otherwise.
- so_u^j is 1 if node u is the source of the jth m-trail and 0 otherwise. Note that only the MN can be the source of an m-trail.
- si_u^j is 1 if node u is the other end node of the jth m-trail and 0 otherwise. If the MN is also the other end node of an m-trail, it will be a cycle.
- ef_{uv}^j is 1 if the jth m-trail, which is either a trail or a cycle, traverses unidirectional link (u,v) en route to its other end node and 0 otherwise.
- er_{uv}^j is 1 if the jth m-trail, which is not a cycle, traverses unidirectional link (u,v) while returning from its other end node to the MN and 0 otherwise.
- e_{uv}^j is 1 if the jth m-trail traverses unidirectional link (u,v) and 0 otherwise.
- eu_{uv}^j is 1 if the jth m-trail traverses undirected link (u,v) and 0 otherwise.
- z_u^j is 1 if the jth m-trail visits node u and 0 otherwise. The variables z_u^j and q_{uv}^j along with predefined constant δ help to keep each m-trail in an ILP solution a single connected component as in [9].
- c^{ij} is 1 if both the ith and jth m-trails traverse any common undirected link and 0 otherwise.
- id_{uv}^{ij} is 1 if both the ith and jth m-trails traverse undirected link (u,v) but the ith m-trail is link-disjoint from the jth m-trail in all other links, and 0 otherwise.
- df_{uv}^j is 1 if the jth m-trail is counted as one of the otherwise link-disjoint m-trails traversing undirected link (u,v), and 0 otherwise.
- es_ψ^j is 1 if the jth m-trail traverses any undirected link of SRLG ψ, and 0 otherwise.

The integer variables are l and n. l is the maximum number of potential collisions for any m-trail in an ILP solution. n is the maximum number of m-trails in an ILP solution traversing through any unidirectional link of the network.

The real variables are q_{uv}^j and α_ψ. q_{uv}^j is a fractional variable. It is defined as *voltage* of the vector $u \to v$ for unidirectional link (u,v). It assumes an arbitrary positive value if the jth m-trail traverses link (u,v) and 0 otherwise. α_ψ is the decimal alarm code of SRLG ψ. Each decimal alarm code must be a positive number.

The specific ILP formulation is provided as follows.

Objective:

$$\text{Minimize}\left\{n + l + r_1 \cdot \sum_j m^j + r_2 \cdot \sum_j \sum_{\forall\,(u,v)\in E} e^j_{uv}\right\}. \quad \text{(A1)}$$

The objective function (A1) aims at minimizing the number of potential collisions between the m-bursts indirectly by minimizing both n and l. The number of m-trails and usage of WLs in the ILP solution are also minimized where each undirected link is traversed by at least $(d+1)$ otherwise link-disjoint m-trails. The constraints are the following.

The constraints (A2)–(A10) and their variables are related with m-trail formation and mostly taken from the method in [9]; please refer to the paper for a thorough explanation of the constraints. A single source node and a single other end node are allowed for each m-trail in an ILP solution. Only the MN can be the source of an m-trail:

$$\sum_{u\in V} so^j_u = m^j, so^j_{MN} = \sum_{u\in V} so^j_u, \sum_{u\in V} si^j_u = m^j, \quad \forall\, j. \quad \text{(A2)}$$

The indices of the m-trails in an ILP solution are the lowest ones. A lower bound of the number of m-trails in an ILP solution helps to find the solution faster:

$$m^j \geq m^{j+1}, \quad \forall\, j : j \leq J-2, \quad \text{(A3)}$$

$$\sum_j m^j \geq d \cdot \lfloor \log_2(E+1) \rfloor. \quad \text{(A4)}$$

The jth m-trail can traverse an undirected link at most once en route to its other end node. A valid m-trail must traverse at least one unidirectional link:

$$ef^j_{uv} + ef^j_{vu} \leq m^j, \quad \forall\,(u,v)\in E : u < v, \quad \forall\, j, \quad \text{(A5)}$$

$$\sum_{\forall\,(u,v)\in E} (ef^j_{uv} + ef^j_{vu}) \geq m^j, \quad \forall\, j. \quad \text{(A6)}$$

The flow conservation defined for each node of the network enforces that each m-trail consists of connected components:

$$\sum_{(u,v)\in E} (ef^j_{uv} - ef^j_{vu}) = (so^j_u - si^j_u), \quad \forall\, u \in V, \quad \forall\, j. \quad \text{(A7)}$$

If any link incident on node u is traversed by the jth m-trail, the node is considered visited by the m-trail. Voltages of the vectors corresponding to on-trail links can be assigned nonzero values. The node voltage constraint (A10) ensures that each m-trail in an ILP solution is a single connected component:

$$z^j_u \geq ef^j_{uv} + ef^j_{vu}, \quad \forall\, u \in V : (u,v)\in E, \quad \forall\, j, \quad \text{(A8)}$$

$$q^j_{uv} \leq ef^j_{uv}, \quad \forall\,(u,v)\in E, \quad \forall\, j, \quad \text{(A9)}$$

$$si^j_u + \sum_{(u,v)\in E} (q^j_{uv} - q^j_{vu}) \geq \delta \cdot z^j_u, \quad \forall\, u \in V, \quad \forall\, j. \quad \text{(A10)}$$

If the MN is the other end node of the jth m-trail, it is a cycle and a trail otherwise:

$$mt^j = m^j - si^j_{MN}, \quad \forall\, j. \quad \text{(A11)}$$

The constraints (A12)–(A14) ensure that if the jth m-trail is a cycle, it can traverse an undirected link at most once. On the other hand, if the m-trail is a trail and traverses an undirected link, it has to traverse the link from both directions:

$$er^j_{uv} \leq ef^j_{vu}, \quad \forall\,(u,v)\in E, \quad \forall\, j, \quad \text{(A12)}$$

$$ef^j_{uv} + er^j_{vu} \leq m^j + mt^j, \quad \forall\,(u,v)\in E, \quad \forall\, j, \quad \text{(A13)}$$

$$m^j + er^j_{vu} \geq mt^j + ef^j_{uv}, \quad \forall\,(u,v)\in E, \quad \forall\, j. \quad \text{(A14)}$$

The jth m-trail can traverse unidirectional link (u,v) either en route to its other end node or while it returns to the MN:

$$e^j_{uv} = ef^j_{uv} + er^j_{uv}, \quad \forall\,(u,v)\in E, \quad \forall\, j. \quad \text{(A15)}$$

The constraint derives the maximum number of m-trails traversing through any unidirectional link of the network:

$$n \geq \sum_j e^j_{uv}, \quad \forall\,(u,v)\in E. \quad \text{(A16)}$$

If both the ith and jth m-trails traverse any common link, c^{ij} will be 1. c^{ij} and c^{ji} always assume the same value:

$$c^{ij} + 1 \geq e^i_{uv} + e^j_{uv}, \quad \forall\,(u,v)\in E, \quad \forall\, i,j : i\langle\rangle j, \quad \text{(A17)}$$

$$c^{ij} = c^{ji}, \quad \forall\, i,j. \quad \text{(A18)}$$

The constraint finds the maximum number of m-trails that traverse common link(s) with an m-trail:

$$l \geq \sum_{\forall\, i : i\langle\rangle j} c^{ij}, \quad \forall\, j. \quad \text{(A19)}$$

If the jth m-trail traverses undirected link (u,v) from either direction en route to its other end node, the link is considered traversed by the m-trail:

$$eu^j_{uv} = ef^j_{uv} + ef^j_{vu}, \quad \forall\,(u,v)\in E : u < v, \quad \forall\, j. \quad \text{(A20)}$$

Constraints (A21) and (A22) identify otherwise link-disjoint m-trail pairs that are traversing each undirected

link (u,v). Here, two m-trails can be considered otherwise link-disjoint if both the m-trails traverse link (u,v) but the ith m-trail is link-disjoint from the jth m-trail in all other undirected links:

$$ld_{uv}^{ij} \le eu_{uv}^{i}, \qquad ld_{uv}^{ij} \le eu_{uv}^{j}, \qquad ld_{uv}^{ij} = ld_{uv}^{ji}, \quad \forall\,(u,v) \in E : u < v, \quad \forall\, i,j : i < j, \tag{A21}$$

$$ld_{uv}^{ij} + eu_{wx}^{i} + eu_{wx}^{j} \le 2, \quad \forall\,(u,v),(w,x) \in E : u < v, \quad \wedge\, w < x \wedge (u,v) \ne (w,x), \quad \forall\, i,j : i < j. \tag{A22}$$

An m-trail traversing the link (u,v) can be considered otherwise link-disjoint from at most d m-trails. At least $(d+1)$ mutually otherwise link-disjoint m-trails should traverse each link of the network:

$$\sum_{i:i\langle\rangle j} ld_{uv}^{ij} = d \cdot df_{uv}^{j}, \quad \forall\,(u,v) \in E : u < v, \quad \forall\, j, \tag{A23}$$

$$\sum_{j} df_{uv}^{j} = (d+1), \quad \forall\,(u,v) \in E : u < v. \tag{A24}$$

If any undirected link (u,v) in SRLG ψ is traversed by the jth m-trail, the SRLG is considered traversed by the m-trail. If no link in SRLG ψ is traversed by the jth m-trail, es_{ψ}^{j} is 0. The decimal alarm code α_{ψ} of SRLG ψ is determined by the set of m-trails traversing the SRLG:

$$es_{\psi}^{j} \ge eu_{uv}^{j}, \quad \forall\,\psi \in \Psi, \quad \forall\,(u,v) \in \psi : u < v, \quad \forall\, j, \tag{A25}$$

$$es_{\psi}^{j} \le \sum_{\forall\,(u,v)\in\psi\,:\,u<v} eu_{uv}^{j}, \quad \forall\,\psi \in \Psi, \quad \forall\, j, \tag{A26}$$

$$\alpha_{\psi} = \sum_{j} 2^{j} \cdot es_{\psi}^{j}, \quad \forall\,\psi \in \Psi. \tag{A27}$$

As a rough indicator of the ILP solving time, the number of binary and integer variables is $O(|E|^{d+1})$. This is because the binary variables are assigned for each link/node and SRLG pair, and we have $|\Psi| \le |E|^{d}$.

Acknowledgment

J. Tapolcai was supported by Hungarian Academy of Sciences (MTA) OTKA grant K108947.

References

[1] J. Lang, Ed., "Link Management Protocol (LMP)," IETF RFC 4204, Oct. 2005.

[2] S. Ahuja, S. Ramasubramanian, and M. Krunz, "Single link failure detection in all-optical networks using monitoring cycles and paths," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1080–1093, Aug. 2009.

[3] J. H. Lee, N. Yoshikane, T. Tsuritani, and T. Otani, "Optical link performance monitoring using extended link management protocol for transparent optical networks," in *Optical Fiber Communication Conf. (OFC)*, San Diego, CA, 2009.

[4] B. Wu and K. L. Yeung, "M^2-CYCLE: An optical layer algorithm for fast link failure detection in all-optical mesh networks," in *Proc. IEEE GLOBECOM*, Dec. 2006, pp. 1–5.

[5] S. Ahuja, S. Ramasubramanian, and M. Krunz, "SRLG failure localization in all-optical networks using monitoring cycles and paths," in *Proc. IEEE INFOCOM*, 2008, pp. 181–185.

[6] H. Zeng, C. Huang, and A. Vukovic, "A novel fault detection and localization scheme for mesh all-optical networks based on monitoring-cycles," *Photon. Netw. Commun.*, vol. 11, no. 3, pp. 277–286, May 2006.

[7] H. Zeng, C. Huang, A. Vukovic, and M. Savoie, "Fault detection and path performance monitoring in meshed all-optical networks," in *Proc. IEEE GLOBECOM*, Dallas, TX, 2004, pp. 2014–2018.

[8] J. Tapolcai, B. Wu, and P.-H. Ho, "On monitoring and failure localization in mesh all-optical networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1008–1016.

[9] B. Wu, P.-H. Ho, and K. Yeung, "Monitoring trail: On fast link failure localization in all-optical WDM mesh networks," *J. Lightwave Technol.*, vol. 27, no. 18, pp. 4175–4185, Sept. 2009.

[10] 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*, Miami, FL, Dec. 2010.

[11] N. Harvey, M. Pătraşcu, Y. Wen, S. Yekhanin, and V. W. S. Chan, "Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs," in *Proc. IEEE INFOCOM*, 2007, pp. 697–705.

[12] J. Tapolcai, L. Rónyai, and P.-H. Ho, "Optimal solutions for single fault localization in two dimensional lattice networks," in *Proc. IEEE INFOCOM*, San Diego, CA, Mar. 2010, pp. 161–165.

[13] 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," *J. Lightwave Technol.*, vol. 29, no. 10, pp. 1597–1606, May 2011.

[14] Y. Wen, V. W. S. Chan, and L. Zheng, "Efficient fault-diagnosis algorithms for all-optical WDM networks with probabilistic link failures," *J. Lightwave Technol.*, vol. 23, no. 10, pp. 3358–3371, Oct. 2005.

[15] J. Tapolcai, P.-H. Ho, L. Rónyai, and B. Wu, "Network-wide local unambiguous failure localization (NWL-UFL) via monitoring trails," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1762–1773, 2012.

[16] B. Wu, P.-H. Ho, J. Tapolcai, and X. Jiang, "A novel framework of fast and unambiguous link failure localization via monitoring trails," in *Proc. IEEE INFOCOM*, San Diego, CA, Mar. 2010, pp. 1–5.

[17] M. L. Ali, P.-H. Ho, B. Wu, J. Tapolcai, and B. Shihada, "Monitoring burst (M-burst)—A novel framework of failure localization in all-optical mesh networks," in *Proc. DRCN*, Krakow, Poland, Oct. 2011.

[18] M. L. Ali, P.-H. Ho, J. Tapolcai, and B. Shihada, "M-burst: A framework of SRLG failure localization in all-optical networks," *J. Opt. Commun. Netw.*, vol. 4, no. 8, pp. 628–638, Aug. 2012.

[19] 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 GLOBEOCOM*, 2010, pp. 1–5.

[20] P. Babarczi, J. Tapolcai, and P.-H. Ho, "Adjacent link failure localization with monitoring trails in all-optical mesh networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 907–920, June 2011.

[21] W. He, P.-H. Ho, B. Wu, and J. Tapolcai, "On identifying SRLG failures in all-optical networks," *Opt. Switch. Netw.*, vol. 10, no. 1, pp. 77–88, 2013.

[22] M. L. Ali, P.-H. Ho, and J. Tapolcai, "SRLG fault localization via M-burst framework," in *Proc. IEEE ICC—Optical Networks and Systems*, Budapest, Hungary, June 2013.

[23] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.

[24] J. M. Aldous and R. J. Wilson, *Graph and Applications: An Introductory Approach*. Springer, 2000.

[25] M. Laguna, "A guide to implementing Tabu search," *Invest. Oper.*, vol. 4, no. 1, pp. 5–25, Apr. 1994.

[26] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. Assoc. Comput. Mach.*, vol. 7, no. 4, pp. 326–329, 1960.

Mohammed L. Ali received his B.Sc. in electrical engineering in 1978 from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh; his B.C.S. (Honours) and M.Sc. in computer science in 2004 and 2007, respectively, from the University of Windsor, Canada; and his Ph.D. in computer science in 2013 from the David R. Cheriton School of Computer Science, University of Waterloo, Canada. He is now a Post-doctoral Fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include fault localization and routing in communication networks.

Pin-Han Ho (M'10) joined the Electrical and Computer Engineering (ECE) Department, University of Waterloo, Waterloo, ON, Canada, in 2002. He is the author/coauthor of more than 200 refereed technical papers and book chapters, and the coauthor of a book on optical networking and survivability. Dr. Ho is the recipient of the Distinguished Research Excellence Award in the ECE Department, University of Waterloo; the Early Researcher Award in 2005; the Best Paper Award at SPECTS 2002 and the IEEE ICC 2005 Optical Networking Symposium; and the Outstanding Paper Award at HPSR 2002.

János Tapolcai (M'09) received his M.Sc. (2000 in technical informatics) and Ph.D. (2005 in computer science) from Budapest University of Technology and Economics (BME), Budapest, and his D.Sc. (2013 in engineering science) from the Hungarian Academy of Sciences (MTA). Currently he is an Associate Professor in the High-Speed Networks Laboratory in the Department of Telecommunications and Media Informatics at BME. His research interests include applied mathematics, combinatorial optimization, optical networks and IP routing, addressing and survivability. He is an author of over 110 scientific publications, and he is the recipient of the Google Faculty Award and Best Paper Award at ICC 2006 and DRCN 2011. He is a TPC member of leading conferences such as IEEE INFOCOM (2012–2014) and is a winner of the MTA Momentum (Lendület) Program.

Suresh Subramaniam (SM) received his Ph.D. in electrical engineering from the University of Washington, Seattle, in 1997. He is a Professor and Interim Chair in the Department of Electrical and Computer Engineering, George Washington University, Washington, DC. His research interests are in the architectural, algorithmic, and performance aspects of communication networks, with particular emphasis on optical networks. He is a coeditor of the books *Optical WDM Networks—Principles and Practice*, *Emerging Optical Network Technologies: Architectures, Protocols, and Performance*, and *Cross-Layer Design in Optical Networks*. Dr. Subramaniam has served on the program committees of several conferences, including INFOCOM, ICC, GLOBEOCOM, OFC, and Broadnets. He served as TPC Co-Chair for INFOCOM 2013, the ICC 2007 Optical Networks Symposium, and LANMAN 2014. He is serving or has served on the editorial boards of *IEEE/ACM Transactions on Networking*, *Optical Switching and Networking*, *KICS Journal of Communications and Networks*, and *Photonic Networks and Communications*. He is a recipient of Best Paper Awards at the ICC 2006 Symposium on Optical Systems and Networks and at the 1997 SPIE Conference on All-Optical Communication Systems.