A Novel Approach for Failure Localization in All-Optical Mesh Networks

János Tapolcai*, Bin Wu†, Pin-Han Ho†, and Lajos Rónyai‡
* Dept. of Telecommunications and Media Informatics, Budapest University of Technology (BME), tapolcai@tmit.bme.hu
† Dept. of Electrical and Computer Engineering, University of Waterloo, Canada, {b7wu,pinhan}@bbcr.uwaterloo.ca
‡ Computer and Automation Research Institute Hungarian Academy of Sciences (MTA SZTAKI), and Inst. of Mathematics, BME, ronyai@sztaki.hu

***Abstract*—Achieving fast and precise failure localization has long been a highly desired feature in all-optical mesh networks. M-trail (monitoring trail) has been proposed as the most general monitoring structure for achieving unambiguous failure localization (UFL) of any single link failure while effectively reducing the amount of alarm signals flooding the networks. However, it is critical to come up with a fast and intelligent m-trail design approach for minimizing the number of m-trails and the total bandwidth consumed, which ubiquitously determines the length of the alarm code and bandwidth overhead for the m-trail deployment, respectively. In this paper, the m-trail design problem is investigated. To gain a deeper understanding of the problem, we first conduct a bound analysis on the minimum length of alarm code of each link required for UFL on the most sparse (i.e., ring) and dense (i.e., fully meshed) topologies. Then, a novel algorithm based on random code assignment (RCA) and random code swapping (RCS) is developed for solving the m-trail design problem. The prototype of the algorithm can be found in [1]. The algorithm is verified by comparing to an Integer Linear Program (ILP) approach, and the results demonstrate its superiority in minimizing the fault management cost and bandwidth consumption while achieving significant reduction in computation time. To investigate the impact of topology diversity, extensive simulation is conducted on thousands of random network topologies with systematically increased network density.**

***Index Terms*—monitoring trails, failure localization, combinatorial group testing**

I. Introduction

In transparent optical networks, failure localization is a very complicated issue that has been extensively investigated [2]–[12]. Due to the lack of optoelectronic regenerators, the impact of a failure propagates without electronic boundary, and a single failure can trigger a large number of redundant alarms [12], [13]. 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 [14], 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. Obviously, 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.

It is obviously not an efficient approach to dedicatedly allocate a monitor for each link. To resolve this situation, the studies in [3]–[6] intorduced the monitoring-cycle (m-cycle) concept, which is a pre-configured loop-back lightpath terminated by an optical power monitor and launched with supervisory optical signals. When any link along the cycle is cut, this interrupts the supervisory lightpath, 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 monitoring structure, [12] 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 was considered in [15], it is called ”permissible probes”. The study has

focused on theoretical proofs and asymptotic bounds, while the strength and flexibility of using tree structures for launching probes has been little explored in possible practical scenarios. More detailed comparison and descriptions of the monitoring structures (e.g. cycles, trails, trees, etc.) can be found in [16].

A number of studies focused on practical implementation of all-optical monitoring using a multi-hop monitoring structure. They aimed at reducing the number of required supervisory wavelength-links, monitoring structures (or active alarms [17]), and/or monitoring locations (MLs) [18]. Here, an ML is defined as a node with at least one active monitor. The study in [18] 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. To distinguish two links l_1 and l_2, a cycle must be disjoint from l_1 while passing through m and l_2. With this method, the number of active alarms will be $O(|E|)$, which is essentially the same as the bound for link monitoring.

The study [17] investigated a redundant alarm reduction problem based on a set of existing working lightpaths, which aimed to minimize the number of active alarms in the network while maintaining single link UFL. An ILP was formulated to solve the problem, and a very fast heuristic was developed to exploit the hierarchical relationship of the monitors. Note that [17] relies on the existing lightpaths for UFL, without allocating any additional supervisory lightpaths for the purpose.

The work in [19] attempted to reduce the number of probes (i.e., the number of monitoring structures or active alarms) sent from some pre-selected MLs for monitoring single or multiple node (or link) failures at the TCP/IP layer. The algorithm first creates a set of candidate probes for every pair of MLs, and the redundant probes are eliminated without affecting the capability for single node UFL hereafter. However, migrating the method to optical networks is subject to a strong limitation: the probes have to follow least-cost paths through the network. Experimental results were obtained on randomly generated topologies with nodal degrees no smaller than 4 for single node UFL. It was observed that the number of probes for UFL increases linearly with the number of nodes in the network. Therefore, an order of $\Omega(n)$ monitoring probes is required in this algorithm, where n is the number of monitored nodes/links for any single failure.

The study in [20] addresses the problem of localizing the faulty processors in a multiprocessor system. They work with the assumption that a monitoring node can monitor the status of any node within a given distance from it, and developed an algorithm for localizing any single node failure. Adapting the scenario to optical networks, nonetheless, is subject to a serious limitation, because in optical networks there is no limit on the number of nodes that can be monitored by a single monitoring node, which was the key design constraint in multiprocessor systems. This holds because there has not been any practical technique or a higher layer protocol that can enable a node to quickly identify the failure of a remote node in a time order of hundreds of milliseconds.

In this paper, we investigate the m-trail design problem for single 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 paper first analytically derives the minimum lengths of alarm codes for ring and fully meshed topologies, respectively. We answer an open question negatively: we construct examples of 2-edge-connected networks with all node degrees at least 3, such that alarm codes of length $O(1) + \log|E|$ are not sufficient for single link UFL. Next, a novel m-trail allocation algorithm is developed for general topologies, which achieves a much better performance in terms of both computation time and solution quality than the ILP in [12]. Extensive simulation is conducted on thousands of randomly generated topologies to investigate the impact of topology diversity on m-trail solutions. We provide an in-depth discussion on the simulation results and draw a number of concluding remarks which position our results on the m-trail design problem.

The rest of the paper is organized as follows. Section II presents the background and problem formulation for m-trail design. Section III provides a comprehensive analysis on the m-trail allocation problem, where the minimum length of alarm code required in a ring and fully meshed topology is obtained as $\lceil |E|/2 \rceil$ and $4 + \lceil \log_2(|E|+1) \rceil$, respectively. Next a construction is proposed as a counter example for the open question on the m-trail design problem for 2-edge-connected topologies with node degrees no smaller than 3. In Section IV, the proposed algorithm for m-trail design is presented. Section V provides simulation results along with ample discussions on our observations. Finally, Section IV concludes the paper.

II. Background

A. Monitoring Trails (M-Trails)

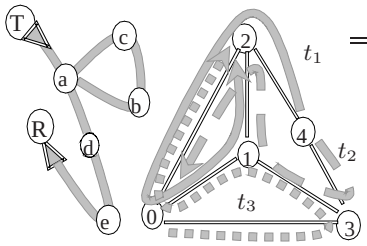
The m-trail concept takes advantage of a monitoring structure in a shape of trails. A trail is a sequence of edges (links) $(v_0, v_1), (v_1, v_2), \ldots, (v_{n-2}, v_{n-1}), (v_{n-1}, v_n)$ in a network $G = (V, E)$, where $(v_i, v_{i+1}) \neq (v_j, v_{j+1})$ if $i \neq j$, and $(v_i, v_{i+1}) \in E$ for $i = 0, \ldots, n-1$. Note that a trail may have repeated nodes but no repeated links. By properly designing a set of m-trails, the network controller can localize a single link failure by collecting the alarm signals issued by the corresponding monitors of m-trails in a timely manner.

An m-trail is a non-simple lightpath with a pair of transmitter and receiver (denoted as T and R, respectively) along with a monitor at the receiver. As shown in Fig. 1(a), the supervisory wavelengths can be pre-cross-connected in either $T \to a \to b \to c \to a \to d \to e \to R$ or $T \to a \to c \to b \to a \to d \to e \to R$. A supervisory optical signal is launched in the m-trail. If any link traversed by an m-trail fails, the optical signal in the supervisory wavelengths is disrupted. After detecting the disruption of the optical signal, the monitor will generate an alarm and broadcast the alarm in the optical network control domain.

Generally, an m-trail solution consists of a set of J m-trails $t_1, t_2, \ldots, t_J$. Upon a single link failure, the monitor on any m-trail traversing the failed link will generate an alarm. At

a remote network entity, an alarm code $[a_1, a_2, \ldots, a_J]$ can be formed after all the flooding alarms are collected, where $a_j = 1$ means that the monitor on m-trail t_j alarms and $a_j = 0$ otherwise. Fig. 1(b) shows a solution with three m-trails t_1, t_2, t_3. If link $(0, 1)$ fails, the monitors on t_1 and t_3 will alarm to produce the alarm code $[1, 0, 1]$. Similarly, if link $(0, 2)$ fails, the monitors on all the three m-trails will alarm and the resulting alarm code is $[1, 1, 1]$. The alarm code table (ACT) in Fig. 1(c) is available at each network routing entity. Thus, the network controller can unambiguously localize a particular link failure by matching the alarm code in the ACT.

Note that the monitoring result will not be affected by having a different pre-cross-connection pattern along the same set of supervisory wavelengths of an m-trail. It is because we only care about whether the supervisory optical signal in the m-trail is disrupted or not, which will yield a single binary digit showing the on/off status of the m-trail.



Link	t_3	t_2	t_1	Dec.
(0,1)	1	0	1	5
(0,2)	1	1	1	7
(0,3)	1	0	0	4
(1,2)	0	1	1	3
(1,3)	1	1	0	6
(2,4)	0	0	1	1
(3,4)	0	1	0	2

(a) m-trail (b) An m-trail solution (c) Alarm code table

Fig. 1. Fast link failure localization based on m-trails.

B. Problem Definition - Deployment of M-Trails

Without loss of generality, the target of m-trail design is to minimize the cost function determined by the weighted sum of *management cost* and *bandwidth cost*. The management cost generally accounts for the fault management complexity in terms of the length of alarm code, 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. The bandwidth cost reflects the additional bandwidth consumption for monitoring, which is measured by way of the *cover length* of an m-trail solution (i.e., the sum of lengths of m-trails in the solution). Here, the *length* of an m-trail is taken as the number of links traversed by the m-trail. The target function adopted in the study is:

$$\begin{aligned} \textit{Total Cost} &= \textit{management cost} + \textit{bandwidth cost} \\ &= \gamma \times (\textit{\# of m-trails}) + \textit{cover length} \end{aligned} \quad (1)$$

The *cost ratio* γ determines the relative importance of management cost and bandwidth cost, which should be defined according to the carrier operational target. In general, the management cost includes not only the cost of the m-trails, but also the efforts on network fault management as explained before. As the wavelength channels are getting cheaper, γ could be chosen much larger than 1 in order to reflect the fact that management cost is much more emphasized.

In order to achieve UFL, each link must be assigned with a unique binary alarm code $[a_1, a_2, \ldots a_J]$, where J is the length of alarm code, and a_j is binary digit, which is 1 if the j^{th} m-trail traverses through this link and 0 otherwise. The m-trail t_j has to traverse through all the links with $a_j = 1$ while avoiding to take any link with $a_j = 0$. Let L_j denote the j^{th} *link set* which contains the set of links with $a_j = 1$. The *link set* L_j *forms an m-trail* t_j if there is a non-simple path that traverses through all the links in L_j but no any other. This is referred to as *m-trail formation*. Here, the theoretical lower bound on J is $\lceil \log_2 (|E| + 1) \rceil$, since there are $|E|$ single failure states plus the no failure state. However, due to the network topology limitation and possibly other design objectives (e.g., the bandwidth consumption limitation), an m-trail solution could take more than $\lceil \log_2 (|E| + 1) \rceil$ m-trails.

In summary, the m-trail design problem aims at finding a set of m-trails in the network with minimized cost as Eq. (1), such that the network controller can unambiguously localize any single link failure by reading the alarm code collected from all the monitors. The alarm code assignments on each link and m-trail formation are two important tasks that should be subject to a joint design. A good m-trail design can only be achieved through a paying attention to both of the two tasks, which makes the m-trail design problem challenging yet interesting.

C. Network Topology Diversity

Network topology diversity imposes a wide impact on the design, development, and deployment of various network algorithms and protocols. Without exception, m-trail solutions are significantly affected by network topologies. Thus, network topology diversity is defined in this subsection.

The study focuses on m-trail design in *2-edge-connected networks*, where any network node has a nodal degree no smaller than 2. The local density of a network topology is defined in terms of average nodal degree. Obviously, the ring and fully meshed topology is the most sparsely and densely connected with an average nodal degree of 2 and $|V| - 1$, respectively, where $|V|$ is the number of nodes. In this case, the number of links is $|V|$ and $|V|(|V| - 1)/2$, respectively.

To simplify the quantification of topology diversity while not loosing much generality, the 2-edge-connected network topology is modeled as a *backbone ring* with a given number of nodes and some additional chords. Fig. 2(a) shows an example of a backbone ring with 14 nodes. The number label at each node is the corresponding nodal degree, which is 2 for each node since there are 2 incident link to each node. When links are added as chords of the ring, the local density is increased accordingly. In the study, the diversity of network topologies is defined and measured in following two ways: the *average nodal degree*, and the total number of nodes with a nodal degree 2 (or *degree-2 nodes*) in the topology. The former measures the amount of added links to the backbone ring, which is the most straightforward way of showing how

densely meshed the topology is. The latter reflects the relative location of the chords of the added links, which measures how *homogeneously* the links are distributed in the network topology. An example is provided as follows. Fig. 2(b) and Fig. 2(c) are two different topologies with 6 added links on the backbone ring. Although with the same average nodal degree (i.e., 2.857), the number of degree-2 nodes is 2 and 9 for Fig. 2(b) and (c), respectively. Intuitively, Fig. 2(c) will require more m-trails for UFL due to the non-homogeneous distribution of the chords.

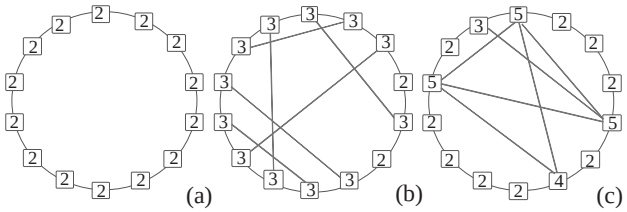


Fig. 2. (a) a backbone ring with 14 nodes; (b) with 2 degree-2 nodes (c) with 9 degree-2 nodes

III. Bound Analysis of M-Trails

Bounds on the minimal number of m-trails to meet the UFL requirement are of interest since they serves indicators on the overhead of network fault management for failure localization. Since an m-trail solution is significantly affected by network topology diversity and connectivity, in the following we will focus on analytical derivation of bounds on the minimum length of alarm code in sparsest (ring) and densest (complete graph) topologies, respectively. Without loss of generality, the subsequent theorems and proofs are developed by only considering the length of alarm code (i.e., the number of m-trails), where γ is equivalently set to much larger than 1. We provide two theorems: the minimum number of m-trails for achieving UFL is no more than $\lceil |E|/2 \rceil$ for rings and $4+\lceil \log_2(|E|+1) \rceil$ for complete graphs. Note that the previous version of the paper [21] provided a construction for every fully meshed topology with at most the $6 + \lceil \log_2(|E|+1) \rceil$ m-trails for achieving the UFL requirement.

The sufficient conditions for a set of network links to form a non-simple path is given by Euler's Theorem:

1) The links must form a connected subgraph;
2) The subgraph has all but up to two (i.e., the two endpoints) odd-degree nodes.

The next Theorem 1 is a simplified version of Theorem 1 in [15].

A. Ring networks

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 lenght of the ring.

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. 3. 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 of 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.

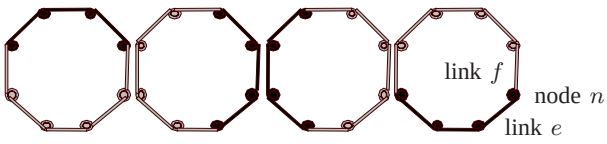


Fig. 3. Optimal M-trail assignment of an 8-node ring.

[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. 3. 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. ■

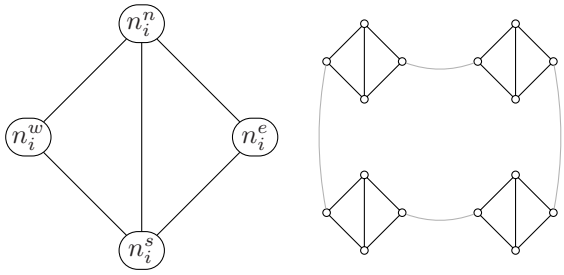
B. Lower bound on the number of m-trails

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 $\log_2(|E|+1)$ due to the binary coding mechanism, which accounts for the fact that it takes $\log_2(|E|+1)$ 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 set of the G_i $i = 1, \ldots n$. 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. 4(b) 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



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

Fig. 4. The structure of each component

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 contains 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 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, where C is a small positive constant. 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 III.C and in [22], respectively), while a ring topology is not (according to Theorem 1). The topology in Fig. 4(b) has $|\bar{E}| = 4$ components although without any degree-2 node, and the structure of the component as illustrated in Fig. 4(a), is the smallest that can found without a degree 2 node. The number of m-trails for the graph of Fig. 4(b) has following lower bound:

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

Eq. 2 holds because each component (as shown in) along with a separating link totally has 6 links, which yields $|E| = 6 \cdot |\bar{E}|$

C. Proposed 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.

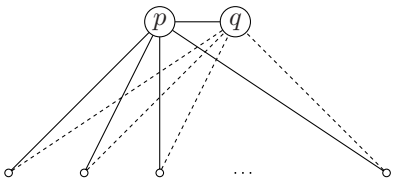


Fig. 5. Subgraph G_p is drawn with solid lines and G_q with broken lines, while G' contains all the rest of the links of the complete graph.

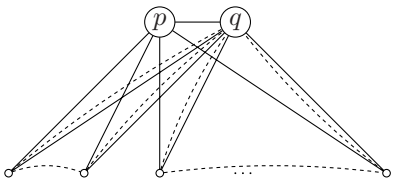


Fig. 6. An example t_{b+3} and t_{b+4}, where t_{b+3} is drawn with solid lines and t_{b+4} with break lines.

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 $b = \lceil \log_2(|E|+1) \rceil$ 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 $v \in V \setminus \{p, q\}$ 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. 5, G_p and G_q are 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_{b+3} and t_{b+4}, to distinguish whether a link of G belongs to G', G_p, or G_q. As shown in Fig. 6, one example to achieve the above is to route the m-trail t_{b+3} through all the links in $G_p \cup G_q$ while t_{b+4} over all the links of G_q (and some links of G'). t_{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_{b+3} is always even except possibly at p and q. Since G_q is a star topology, the routing of t_{b+4} needs some links from G' until the Euler property is met. An example of such link set is the edge set in G' of a perfect matching.

Note that t_{b+4} is used to distinguish the links in G_p from those in G_q, and t_{b+3} is to distinguish links in G' from those in G_p or in G_q. Therefore with t_{b+3} and t_{b+4}, the 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 lenght $\lceil \frac{b+1}{2} \rceil$ bits

are generated for the links in E_p. This can be done because

$$2^{\lceil \frac{b+1}{2} \rceil} \geq 2^{\frac{b+1}{2}} = \sqrt{2 \cdot 2^b} \geq \sqrt{2|E|} = \sqrt{2\frac{|V|(|V|-1)}{2}} > \sqrt{(|V|-1)^2} = |V|-1. \quad (3)$$

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:	$t_1 \dots t_{\lceil \frac{b+1}{2} \rceil}$	$t_{\lceil \frac{b+3}{2} \rceil} \dots t_{b+2}$	t_{b+3}	t_{b+4}
for links in E_p	core code	$x \dots xx$	1	0

where x denotes the yet undefined bits.

Step (4) Next unique non-zero binary codes with $\lceil \frac{b+1}{2} \rceil$ 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 $\lceil \frac{b+1}{2} \rceil$ 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:	$t_1 \dots t_{\lceil \frac{b+1}{2} \rceil}$	$t_{\lceil \frac{b+3}{2} \rceil} \dots t_{b+2}$	t_{b+3}	t_{b+4}
for links in E_q	$x \dots xx$	core code	1	1

Step (5) Unique non-zero codes with $b+1$ bits are generated as the *core codes* for the links in E'. Note that this can easily be done since $|E| < 2^b$. 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 [23], [24]), 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 $1, \dots, \lceil \frac{b+1}{2} \rceil$. Note that according to Eq. (3), at least $|V|-1$ such codes exist. Similarly, for each link of H_2, "1" is assigned to each bit at the bit positions $\lceil \frac{b+3}{2} \rceil, \dots, b+1$. The format of the codes for the links of E' is given as follows.

m-trails:	$t_1 \dots t_{\lceil \frac{b+1}{2} \rceil}$	$t_{\lceil \frac{b+3}{2} \rceil} \dots t_{b+1}$	t_{b+2}	t_{b+3}	t_{b+4}
for links in H_1	$11 \dots 1$	code fragment		0	x
H_2	code frag.	$11 \dots 1$	1	0	x
$E' \setminus H_1 \cup H_2$	core code in $b+1$ bits		x	0	x

Step (6) After Step (2) - Step (5), we can identify the link set L_j, $\forall 1 \leq j \leq b+2$, 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 $j \leq \lceil \frac{b+1}{2} \rceil$, and (v, p) is added to L_j if $\lceil \frac{b+1}{2} \rceil < j \leq b+2$. 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, $(p, q) \in L_j$ 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 $4 + \lceil \log_2(|E|+1) \rceil$ 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 later, 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 $(b+3)$-th and $(b+4)$-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

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

hence for $1 \leq j \leq \lceil \frac{b+1}{2} \rceil$ the edges of G_q, while for $\lceil \frac{b+3}{2} \rceil \leq j \leq b+2$ 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 $L_j, \forall 1 \leq j \leq (b+4)$, will form a valid m-trail.

With all the above, we proved that the proposed construction has each link coded with $(b+4)$ bits. This gives $(b+4)$ 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 [15] by Algorithm 1, and thus it improves the $O(\log_2|E|)$ construction (Theorem 2 of [15]) to $O(1) + \log_2|E|$.

IV. Algorithm For M-Trail Solution

A novel algorithm for achieving a fast and efficient m-trail design in general topologies is introduced in this section. 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. (1). In our design, 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. 7 shows a flowchart of the proposed algorithm. At the beginning, an ACT 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 ACT is evaluated by Eq. (1) (which will be further elaborated in subsection IV-A). 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 IV-B). In every cycle, a new ACT (denoted as ACT_{new}) is generated and the corresponding cost C_{ACT_new} is evaluated in Step (2). If the cost of ACT_{new} (denoted as C_{ACT_new}) is smaller than (or equal to) that of the cost of previous ACT (denoted C_{ACT}) as in Step (2), the algorithm starts the next greedy cycle by replacing the old ACT with the new one (i.e., $ACT \leftarrow ACT_{new}$) and performing RCS as denoted as $ACT_{new} \leftarrow \Psi_{RCS}(ACT_{new})$ 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).

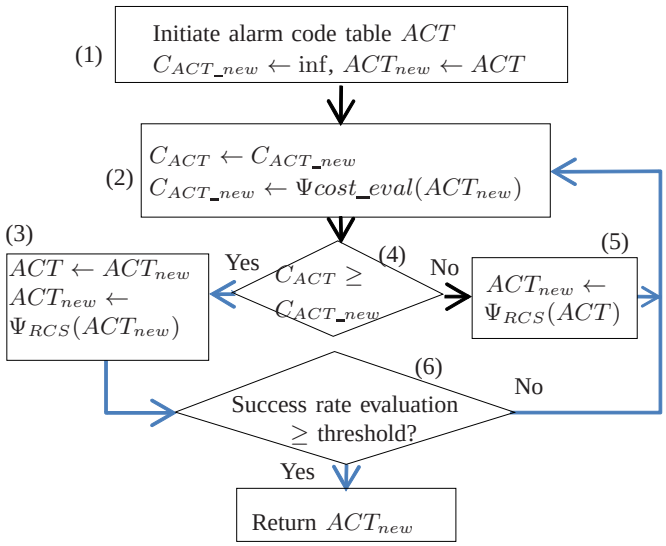


Fig. 7. The flowchart for the proposed heuristic algorithm.

A. M-trail Formation

This subsection introduces the basic idea of our m-trail formation mechanism, where Eq. (1) is used to evaluate the ACT in each greedy cycle in Fig. 7 such that the greedy cycle can possibly converge and yield a set of feasible m-trails with high quality.

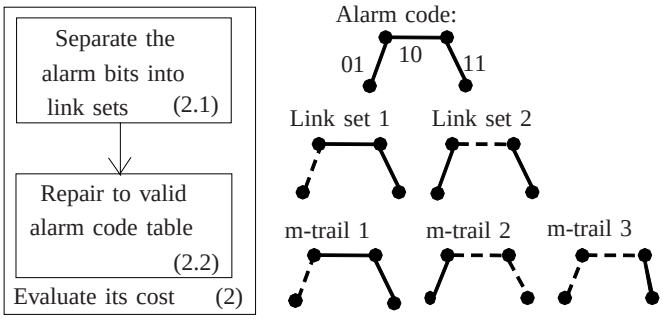


Fig. 8. An illustration of the cost evaluation method of an alarm code table of the 4 node line graph.

Fig. 8 elaborates Step (2) of Fig. 7 through an example by considering a simple three-link topology. Initially, a 2-bit long alarm code is assigned to each link. The formation of the j^{th} m-trail has to take all the links with $a_j = 1$ (which belong to L_j) as shown in Step (2.1) of Fig. 8. The ideal situation is that an ACT with J bits yields exactly J link sets, which 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 necessarilyy 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 2 of Fig. 8), it will possibly be constructed as a union of multiple m-trails or cycles according to the following Lemma.

Lemma 1: 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. 8.

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.

B. 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 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 [1].

C. 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. 9(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. 9(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. 7. The effectiveness and efficiency of the proposed algorithm will be further demonstrated in the next section.

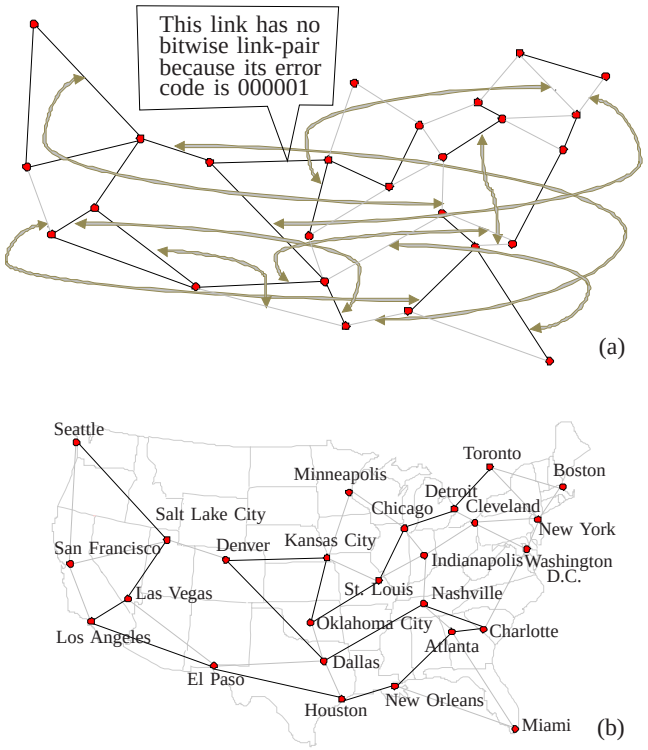


Fig. 9. Link set 6 (a) after random coding (b) after greedy random code swapping for link set 6 in the ACT.

V. Simulation

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 [12]; and (2) investigate impacts of topology diversity on m-trail solutions.

A. 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 [12]. 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. 10 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. 10 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.

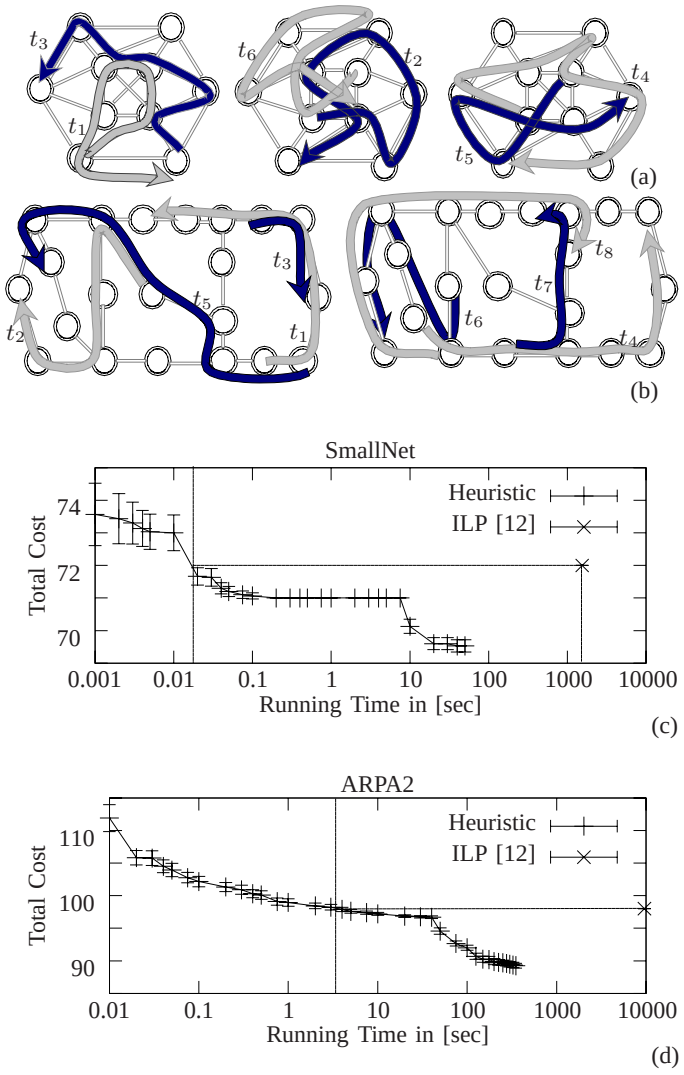


Fig. 10. The best heuristic solutions for SmallNet with cost of 69 and ARPA2 with cost of 87. In comparisons the ILP resulted 72 with 4.17% gap-to-optimality for and SmallNet and 98 with 20.41% for ARPA2.

B. 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. 11 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. 11 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. 11(a) and Fig. 11(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. 11(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. (1).

Fig. 11(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.

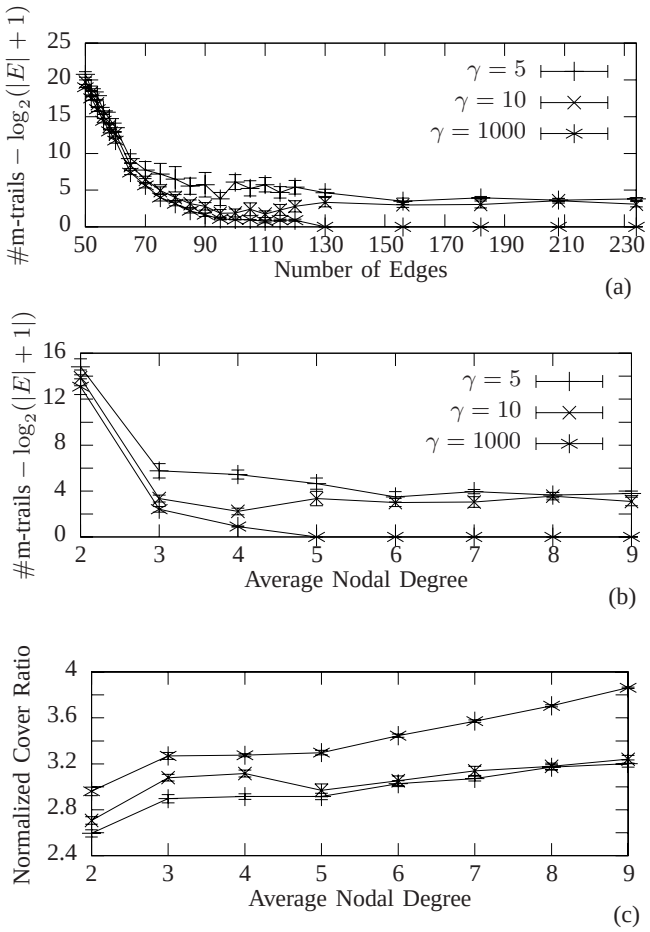


Fig. 11. Simulation on topologies with 50 nodes. (a), (b): normalized length of alarm code versus number of links (or the average nodal degree); and (c): normalized cover ratio versus average nodal degree.

Fig. 12 shows the m-trail solutions with different numbers

of nodes in the network topologies. Fig. 12(a) and (b) shows the length of the alarm code versus the number of links. Clearly, when γ = 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.

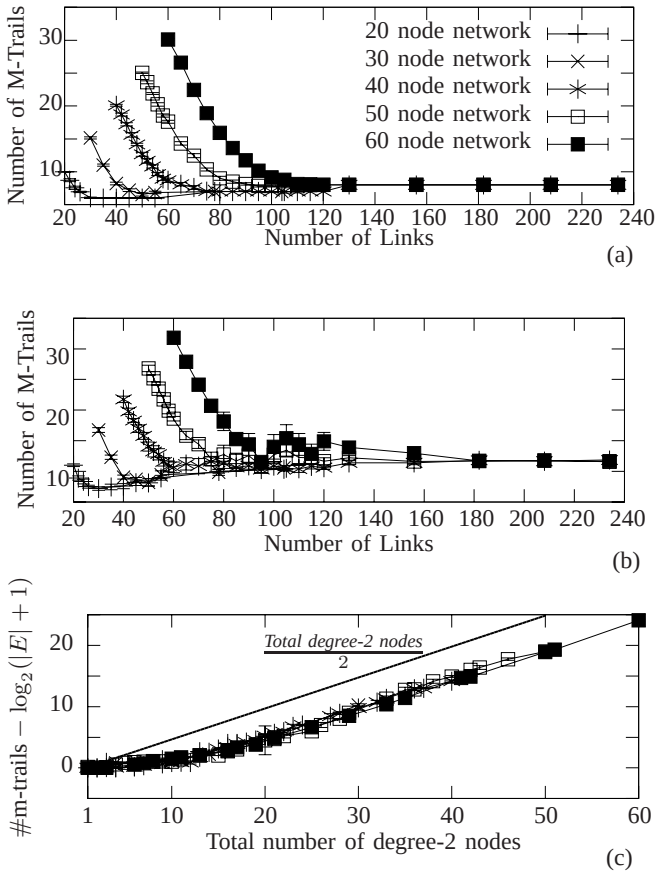


Fig. 12. The length of alarm code versus the number of added links for topologies of 20, 30, 40, 50, and 60 nodes. (a) $\gamma = 1000$ and (b) $\gamma = 5$.

Fig. 12(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 to the number of degree-2 nodes in a topology. We found that the following relationship holds and 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:

$$\#\text{m-trail} \quad \approx \quad \lceil \log_2(|E|+1) \rceil \; + \; \frac{\#\text{degree-2 nodes}}{2} \qquad (4)$$

VI. Concluding Remarks

As a generalization of all the previously reported all-optical monitoring structures, Monitoring-Trail (m-trail) has been an effective approach for achieving unambiguous failure localization (UFL) under any single failure. The paper investigated the m-trail design problem by first obtaining the minimum length of alarm code (or equivalently, the minimum number of m-trails) for ring and fully meshed topologies. We have also investigated an open question: whether a 2-edge-connected topology without any degree-2 node can always achieve an m-trail solution with $O(1)+\log_2(|E|+1)$ m-trails. To obtain fast and high-quality m-trail solutions, a novel algorithm based on random code assignment (RCA) and random code swapping (RCS) was introduced. Extensive simulation was conducted to verify the proposed algorithm and obtain insights on the m-trail design problem. We have seen that the proposed algorithm can achieve significantly better performance compared to our previously developed enumeration-free ILP model. The new algorithm uses much less computation time without losing solution quality.

With the proposed algorithm, the impact on m-trail solutions of topology diversity and cost ratio γ were analyzed by conducting experiments on thousands of different topologies. Our observations are summarized as follows:

1) In case the bandwidth cost is not emphasized (i.e., γ is very large), the minimum length of alarm code decreases as network density is increases. The minimum length of alarm code approaches $\lceil \log_2(|E|+1) \rceil$, and the normalized cover ratio grows mildly, when most nodes have a nodal degree larger than 2, or generally when the average nodal degree reaches 3∼4.
2) When the emphasis on management cost and bandwidth cost is comparable (i.e., γ is close to 1), the impact due to the number of degree-2 nodes becomes more significant. Meanwhile, the normalized cover ratio is getting smaller as the average nodal degree increases due to a smaller γ.
3) The minimum length of the alarm code can be generally approximated by way of the total number of degree-2 nodes in the topology as shown in Eq. (2).

Acknowledgements

This work has been partially supported by Discovery Grant, National Science and Engineering Research Council (NSERC), Canada, High Speed Network Laboratory (HSNLab), the Hungarian National Research Fund, and the National Office for Research and Technology (Grant Number OTKA NK 72845, K77476, K77778, and 67651). J. Tapolcai was supported by Magyary Zoltán postdoctoral program.

REFERENCES

[1] J. Tapolcai, "Web page on m-trail/tree design: simulation environments, examples and technical reports," http://opti.tmit.bme.hu/∼tapolcai/mtrail.

[2] C. Mas, I. Tomkos, and O. Tonguz, "Failure Location Algorithm for Transparent Optical Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 8, pp. 1508–1519, 2005.

[3] H. Zeng and C. Huang, "Fault detection and path performance monitoring in meshed all-optical networks," in *IEEE GLOBECOM '04*, vol. 3, 2004, pp. 2014–2018.

[4] 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.

[5] 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.

[6] B. Wu and K. Yeung, "M^2-CYCLE: an Optical Layer Algorithm for Fast Link Failure Detection in All-Optical Mesh Networks," in *IEEE GLOBECOM '06*, 2006, pp. 1–5.

[7] B. Wu, K. Yeung, and P.-H. Ho, "Monitoring Cycle Design for Fast Link Failure Localization in All-Optical Networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 27, no. 10, pp. 1392–1401, 2009.

[8] C. Li, R. Ramaswami, I. Center, and Y. Heights, "Automatic fault detection, isolation, and recovery in transparentall-optical networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 15, no. 10, pp. 1784–1793, 1997.

[9] S. Stanic, S. Subramaniam, H. Choi, G. Sahin, and H. Choi, "On monitoring transparent optical networks," in *Proc. International Conference on Parallel Processing Workshops (ICPPW '02)*, 2002, pp. 217–223.

[10] Y. Wen, V. Chan, and L. Zheng, "Efficient fault-diagnosis algorithms for all-optical WDM networks with probabilistic link failures," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, pp. 3358–3371, 2005.

[11] 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 *IEEE GLOBECOM '02*, vol. 3, 2002, pp. 2676–2680.

[12] B. Wu, P.-H. Ho, and K. Yeung, "Monitoring Trail: On Fast Link Failure Localization in WDM Mesh Networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 27, no. 23, Dec. 2009.

[13] M. Maeda, "Management and control of transparent optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1008–1023, 1998.

[14] P. Demeester, M. Gryseels, A. Autenrieth, C. Brianza, L. Castagna, G. Signorelli *et al.*, "Resilience in multilayer networks," *IEEE Communications Magazine*, vol. 37, no. 8, pp. 70–76, 1999.

[15] 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 *IEEE INFOCOM*, 2007, pp. 697–705.

[16] 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," *to appear in the First issue 2011 IEEE Communications Surveys & Tutorials*, 2011.

[17] S. Stanic, S. Subramaniam, G. Sahin, H. Choi, and H. A. Choi, "Active monitoring and alarm management for fault localization in transparent all-optical networks," *IEEE Transactions on Network and Service Management*, vol. 7, no. 2, pp. 118–131, 2010.

[18] S. Ahuja, S. Ramasubramanian, and M. Krunz, "Single link failure detection in all-optical networks using monitoring cycles and paths," *accepted for publication in IEEE/ACM Transactions on Networking*, 2009, http://www.ece.arizona.edu/ srini/Publications.php.

[19] M. Brodie, I. Rish, and S. Ma, "Optimizing probe selection for fault localization," in *EEE International Workshop on Distributed Systems: Operation and Management*, Nancy, France, 2001.

[20] M. Karpovsky, K. Chakrabarty, and L. Levitin, "On a new class of codes for identifying vertices in graphs," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 599–611, 1998.

[21] 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.

[22] J. Tapolcai, L. Rónyai, and P.-H. Ho, "Optimal solutions for single fault localization in two dimensional lattice networks," in *Proc. IEEE INFOCOM Mini-Symposium*, San Diego, CA, USA, 2010.

[23] E. Lucas, *Recreations Mathematiques*. Gauthier-Villars, Paris, 1893.

[24] B. Alspach, "The wonderful walecki construction," *Bull. Inst. Combin. Appl*, vol. 52, pp. 7–20, 2008.

János Tapolcai received his M.Sc. ('00 in Technical Informatics), and Ph.D. ('05 in Computer Science) degrees in Technical Informatics from Budapest University of Technology and Economics (BME), Budapest, Hungary. Currently he is an Associate Professor at the High-Speed Networks Laboratory at the Department of Telecommunications and Media Informatics at BME. His research interests include applied mathematics, combinatorial optimization, linear programming, linear algebra, routing in circuit switched survivable networks, availability analysis, grid networks, and distributed computing. He has been involved in several related European and Canadian projects. He is an author of over 40 scientific publications, and is the recipient of the Best Paper Award in ICC'06.

Bin Wu received the B.Eng. degree from Zhe Jiang University, Hangzhou, China, in 1993, M.Eng. degree from University of Electronic Science and Technology of China, Chengdu, China, in 1996, and PhD degree from the University of Hong Kong, Hong Kong, in 2007. During 1997-2001, he served as the department manager of TI-Huawei DSP co-lab in Huawei Tech. Co. Ltd, Shenzhen, China. Currently he is a postdoctoral research fellow at the University of Waterloo, Waterloo, Canada.

Pin-Han Ho received his B.Sc. and M.Sc. degrees from the Electrical and Computer Engineering, Department of National Taiwan University in 1993 and 1995, respectively. He started his Ph.D. studies in 2000 at Queen's University, Kingston, Ontario, Canada, focusing on optical communications systems, survivable networking, and QoS routing problems. He finished his Ph.D. in 2002, and joined the Electrical and Computer Engineering Department at the University of Waterloo as an assistant professor in the same year. He is the author/co-author of more than 100 refereed technical papers and book chapters, and the co-author of a book on optical networking and survivability. He is the recipient of the Distinguished Research Excellence Award in the ECE Department at the University of Waterloo, the Early Researcher Award in 2005, the Best Paper Award at SPECTS '02 and the ICC '05 Optical Networking Symposium, and the Outstanding Paper Award in HPSR '02.

Lajos Rónyai is a research professor with the Informatics Laboratory of the Computer and Automation Institute of the Hungarian Academy of Sciences. He leads a research group there which focuses on theoretical computer science and discrete mathematics. He is also a full professor at the Mathematics Institute of the Budapest University of Technology and Economics. He received his PhD in 1987 from the Eötvös Loránd University, Budapest. His research interests include efficient algorithms, complexity of computation, algebra, and discrete mathematics.