On Monitoring and Failure Localization in Mesh All-Optical Networks

János Tapolcai*, Bin Wu†, and Pin-Han Ho†
* Dept. of Telecommunications and Media Informatics, Budapest University of Technology, tapolcai@tmit.bme.hu
† Dept. of Electrical and Computer Engineering, University of Waterloo, Canada, {b7wu,pinhan}@bbcr.uwaterloo.ca

***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 flooded in 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 totally consumed bandwidth, which ubiquitously determines the length of alarm code and bandwidth overhead for the m-trail deployment, respectively. In this paper, the m-trail design problem is investigated. To gain deeper understanding of the problem, we firstly conduct a bound analysis on the minimum length of alarm code required for UFL. 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 algorithm prototype can be found in [1]. The algorithm is verified by comparing with an Integer Linear Program (ILP), 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 connectivity. Lastly, we provide abundant discussions and interesting conclusive remarks that position our discoveries.**

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 a 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 management cost of 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 be jointly considered.

Conventional link-based monitoring scheme requires one monitor at each link, which requires $O(|E|)$ optical monitors and a length of alarm code as $|E|$, where $|E|$ is the number of network links. To make the solution more scalable, the studies in [3]–[6] investigated monitoring-cycle (m-cycle), which is a pre-configured optical loop-back connection terminated by a monitor and launched with supervisory optical signals. When any link along the cycle is cut, the failure will be detected by the monitor, and the monitor will issue an alarm to the rest of the network. Instead of using dedicated lightpaths for fault monitoring, the studies in [15], [16] proposed to construct one or a number of trees or simple cycles or simple paths that are rooted at one or a number of monitoring locations, where a probe is proactively launched in each tree/cycle/path to identify any single component failure. Although interesting, the two studies have not come up with a systematic approach that can optimally identify the monitoring resources. Also, the monitoring structure is limited in the shapes of trees, simple cycles, and simple paths. Similarly, the study in [17] has focused on SRG failure localization, where an Integer Linear Program (ILP) using cycle/path enumeration and a heuristic based on graph decomposition were introduced.

To erase the limitation on the monitoring structure, our previous work [12] introduced the concept of Monitoring-Trail (m-trail), where an optimal model based on an enumeration-free ILP was introduced for m-trail design. M-trail is proved to yield much better performance by employing monitoring resources in a 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.

In this paper, we investigate the m-trail design problem and aim at obtaining deeper understanding and insights in the problem. Firstly, we analytically derive the minimum length of alarm code for ring and fully meshed topologies, respectively. The derivation is based on a deterministic linear algorithm of m-trail design and random code assignment (RCA), which serve as the foundation of the subsequent algorithm develop-

ment for general topologies. Based on similar techniques, a novel algorithm is developed for general topologies, which is proved to achieve much better performance in terms of both computation time and solution quality than that by solving 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 conclusive remarks which position our discoveries 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 $6 + \lceil \log_2 (|E| + 1) \rceil$, respectively. In Section IV, the proposed algorithm for m-trail design is presented. Section V provides simulation results along with abundant discussions on our observations. Finally, the paper concludes in Section IV.

II. Background

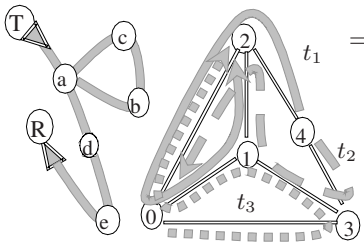
A. Briefs on Monitoring Trails (M-Trails)

The m-trail concept takes advantage of a monitoring structure in a shape of trails, which generalizes all the previously reported counterparts. In brief, an m-trail can traverse a node multiple times. By allocating a sufficient number of m-trails, a routing entity in the network can localize a single 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 in optical networks 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 $T \rightarrow a \rightarrow b \rightarrow c \rightarrow a \rightarrow d \rightarrow e \rightarrow R$ or $T \rightarrow a \rightarrow c \rightarrow b \rightarrow a \rightarrow d \rightarrow e \rightarrow R$. A supervisory optical signal is launched in the m-trail. If any link traversed by an m-trail fails, optical signal in the supervisory wavelengths is disrupted. After detecting the disruption of the optical signal, the monitor will generate an alarm and flood 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 $[a_J, \ldots, a_2, a_1]$ 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 equipped in each network routing entity, which maintains all the possible alarm codes that could be received at a network entity. Thus, the network entity 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 linear combination of *monitoring cost* and *bandwidth cost*. The monitoring 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 length of each m-trail 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:

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

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

In order to achieve UFL, each link must be assigned with a unique binary alarm code $[a_J, a_{J-1}, \ldots\ldots a_1]$, 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 $a_j = 1$; thus, to form t_j, we have to find 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 $\lceil \log_2 (|E| + 1) \rceil$; 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 a network entity can unambiguously localize any single failure event by reading the alarm code collected from all the monitors. The alarm code assignment on each link and m-trail formation are two important tasks that should be subject a joint design. A good m-trail design can only be achieved through a manipulative interplay 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-connected networks, where any network node has a nodal degree no smaller than 2. The connectivity 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 without loss of generality, a *backbone ring* is defined with a given number of nodes. Fig. 2(a) shows an example of a backbone ring with 14 nodes. The number labeled 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 connectivity is increased accordingly. In the study, the diversity of network topologies is defined and measured in following two folds: the average nodal degree, and the total number of nodes with a nodal degree 2 (or termed *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. Obviously, Fig. 2(c) will require more m-trails for UFL due to the non-homogeneous distribution of the chords.

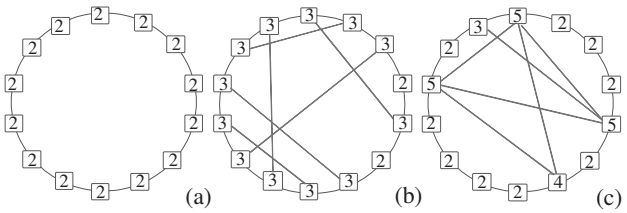


Fig. 2. (a) a backbone ring with 14 nodes; (b) with 2 degree-2 nodes (c) with 9 degree-2 nodes

III. Bound Analysis of M-Trails

The bounds on the minimal number of m-trails to meet the UFL requirement are of interest since they can give us a clear picture 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 followings we will focus on analytical derivation of the bounds on the minimum length of alarm code in ring and fully meshed 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 for a ring and fully meshed network that the minimum number of m-trails for achieving the UFL requirement is no more than $\lceil |E|/2 \rceil$ and $6 + \lceil \log_2 (|E| + 1) \rceil$, respectively. To support the two theorems, a number of necessary lemmas are introduced accordingly.

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 minimum number of m-trails for a ring topology is derived as $\lceil |E|/2 \rceil$ given in Lemma 1 which is a simplified version of Theorem 1 of [15].

Lemma 1: A ring network of more than 4 nodes needs $\lceil |E|/2 \rceil$ m-trails for UFL.

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

[Proof of claim (1)] Let e and f be two links with a common adjacent node n, 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 n has a nodal degree of two, this can only happen if an m-trail terminates at node n. It is clear that for a ring topology, a number of $\lceil |E|/2 \rceil$ 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 such that any failed link can be unambiguously localized according to the status of the m-trails. [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. In case the ring has an odd number of nodes, the last m-trail must be 2-

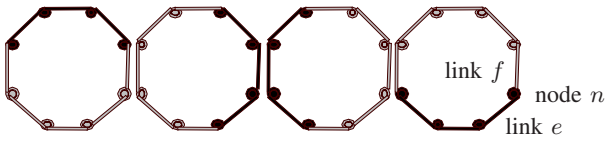


Fig. 3. Optimal M-trail assignment of an 8-node cycle.

hop in length. Thus, the network needs up to $\lceil |E|/2 \rceil$ m-trails for achieving UFL. ■

Lemma 1 can be extended to the scenario of general two-connected topologies, as shown in Theorem 1. Obviously, a ring topology takes the maximum number of m-trails to achieve UFL among all the topologies with the same number of nodes.

Theorem 1: In any two-connected network, the UFL requirement can always be achieved with no more than $\lceil |E|/2 \rceil$ m-trails.

Proof: The theorem is proved with a deterministic polynomial time construction that generates a valid m-trail solution for a two-connected topology with at most $\lceil |E|/2 \rceil$ m-trails.

Firstly, we define a *circuit* that goes through every link of the topology, and some links may be passed twice. The shortest circuit can be found by solving the *Chinese postman problem* [18]. The links that are passed twice by the circuit are called *duplex links*. Let a *tour order* of the links be defined such that every link is visited and labeled one after the other along the circuit starting from an arbitrary node n. Meanwhile, 3-hop m-trails are allocated in the tour such that each pair of consecutively allocated m-trails overlaps by a single hop. During the tour, the length of an m-trail is increased by one hop when the m-trail passes through any duplex link that has been already labeled. Since each link is only labeled once, totally $\lceil |E|/2 \rceil$ m-trails will be sufficient such that the failure of any labeled link can be unambiguously identified by the m-trail solution. ■

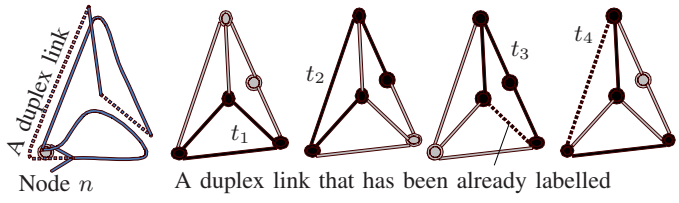


Fig. 4. The m-trails based on the close trail decomposition of a graph.

Theorem 2 gives an upper bound on the minimum number of m-trails required for UFL in a fully meshed topology. The theorem is proved by way of a deterministic polynomial time construction that generates a valid m-trail solution for a fully mesh topology with $6 + \lceil \log_2 (|E| + 1) \rceil$ m-trails. The proof of the theorem also serves as a foundation for the development of the proposed algorithm, which will be introduced in Section IV.

Theorem 2: A fully meshed topology needs no more than $6 + \lceil \log_2 (|E| + 1) \rceil$ m-trails to achieve UFL.

Proof: Based on Theorem 1, a fully meshed topology with no more than 7 nodes can always be solved within $6 + \lceil \log_2 (|E| + 1) \rceil$ m-trails due to the fact that $\lceil \frac{21}{2} \rceil \leq 6 + \lceil \log_2 22 \rceil = 11$. With a fully meshed topology with 8 nodes or more, at least 4 disjoint Hamiltonian cycles can be identified in the topology [19]. Strategically, the topology is decomposed into three link-disjoint subgraphs denoted as $G_1(E_1, V)$, $G_2(E_2, V)$, and $G_3(E_3, V)$, such that each G_2 and G_3 consists of a union of two Hamiltonian cycles, respectively, while G_1 contains all the rest of the links in the topology. In this case, the number of links in G_1 is $|E_1| = |E| - 4 \cdot (|V| - 1)$, and $|E_2| = |E_3| = 2 \cdot (|V| - 1)$ for each G_2 and G_3.

We firstly add 2 m-trails for identifying the subgraph subject to a link failure among the three. With t_2' and t_3', the overall UFL can be sufficiently achieved by ensuring UFL separately in each of the three subgraphs. The existence of the 2 m-trails is guaranteed due to the special structure of G_2 and G_3, where all links in each subgraph can be covered with a single m-trail, denoted as t_2' and t_3', respectively.

Two link sets of different subgraphs can be *merged* into a single link set. Without loss of generality, the two link sets are assumed to be in G_1 and G_2, denoted as $L_j^{G_1}$ and $L_k^{G_2}$, respectively; and the merged link set is denoted as $L_{j,k}^{G_{1,2}} = L_j^{G_1} \cup L_k^{G_2}$. With t_2' and t_3', $L_{j,k}^{G_{1,2}}$ is not responsible in failure localization of G_3. Thus, the proof of the theorem requires the proof of the following two claims: (1) No less than $\lceil \log_2 |E_G + 1| \rceil$ link sets are required for UFL in a topology of E_G links; and (2) any merged link set between two subgraphs can be shaped into a valid m-trail by possibly taking the links of the third subgraph. The proof of the two claims is provided in the appendix as Lemma 3, Lemmas 4. Note that the proof of claim (2) takes advantage of the fact that any link of G_3 can be freely taken in the m-trail formation on $L_{j,k}^{G_{1,2}}$.

With the two claims, the minimum required m-trails for a fully meshed topology is the number of merged link sets plus two (due to t_2' and t_3'), where each link set of G_1 is merged with a link set in G_2 or G_3 as described in Lemmas 4. Note that G_1 is coded on almost twice as many bits as G_2 and G_3. In other words, the number of link sets of G_1 is almost as many as the link sets of both G_2 and G_3. Clearly, G_1 requires no more than $\lceil \log_2 (|E| + 1) \rceil$ m-trails according to Lemma 3, and a maximum of four m-trails are required after merging the link sets according to Eq. (3). Thus, in addition to t_2' and t_3', there will be totally $6 + \lceil \log_2 (|E| + 1) \rceil$ m-trails required for UFL, which completes the proof. ■

Note that the abovementioned construction is a special case of the problem addresses 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 a general topology 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 in general topologies. With RCA, it takes $|E|$ unique alarm codes to be randomly assigned to each link one after the other at the beginning and is kept in an alarm code table (ACT), which leads to $\lceil \log_2(|E|+1) \rceil$ link sets. The algorithm then performs m-trail formation by examining the connectivity of the links in 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 due to the stipulation that 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 *strong locality constraint* (SLC), which is an important feature of our design in making the algorithm simpler and running faster.

Fig. 5 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 has been done without getting a smaller cost at Step (4).

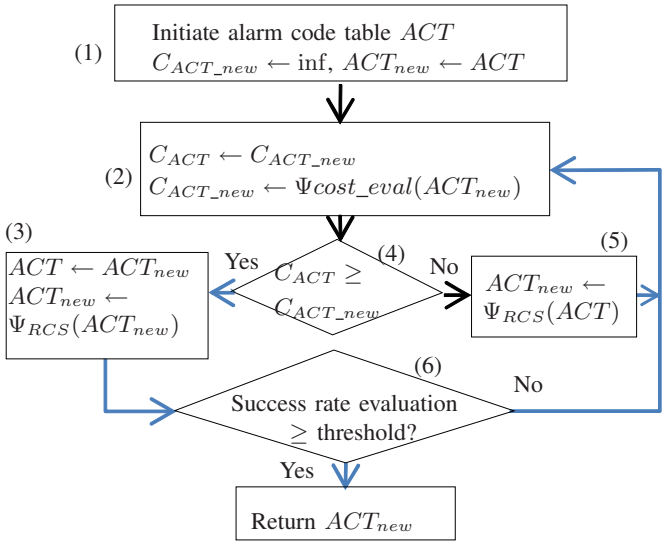


Fig. 5. The flowchart for the proposed heuristic algorithm.

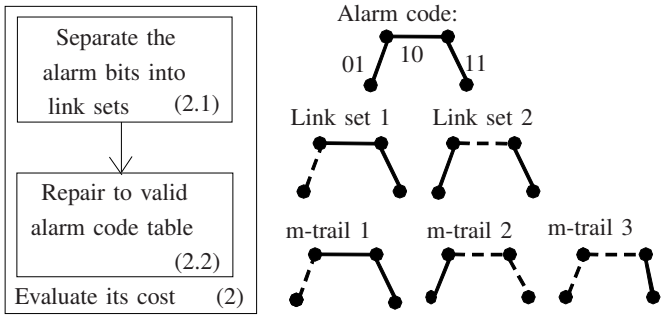


Fig. 6. An illustration of the cost evaluation method of an alarm code table of the 4 node line graph.

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. 5 such that the greedy cycle can possibly converge and yield a set of feasible m-trails with high quality.

Fig. 6 elaborates Step (2) of Fig. 5 through an example by considering a simple three-link topology. Initially, an alarm code with 2-bit long 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. 6. 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 exclusively by a non-simple path. In other words, the link set can have maximally two nodes with an odd nodal degree according Euler's theorem. Checking the validity of a link set for 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. 6), it will possibly be constructed as a union of multiple m-trails or cycles as described in Lemma 2 (in appendix), which 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 $\#odd(C_{ij})$), it can be decomposed into $\max(1, \#odd(C_{ij})/2)$ m-trails. This is also exemplified in Step (2.2) of Fig. 6.

In case m-trail t_{J+1} is newly identified, the alarm code, which was originally J bits in length, is appended with an additional bit, denoted as a_{J+1}. Thus, a_{J+1} is set to 1 for all links traversed by t_{J+1}, and 0 otherwise. Similarly, when t_j merges with t_k to form a new m-trail t'_j, a_j is set to 1 for all links taken by $t_j \cup t_k$. The k^{th} bit of alarm code of each link is removed accordingly.

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 on the strong locality constraint (SLC) on the swapping mechanisms in each link set. It stipulates when RCS is on L_j for m-trail formation, 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 that for the others. Such a mechanism yields merits in terms of easy implementation and 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 numbers 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 further provide an example as follows to show how RCS is performed. The 26-node US network with 42 links are uniquely and randomly coded in 6 bits at the beginning. Fig. 7(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. 7(a) shows each link-pair at L_6 by an arrow. For example, (Atlanta, Charlotte) link and (Indianapolis, Cleveland) link 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. 5. The effectiveness and efficiency of the proposed algorithm will be further demonstrated in the next section.

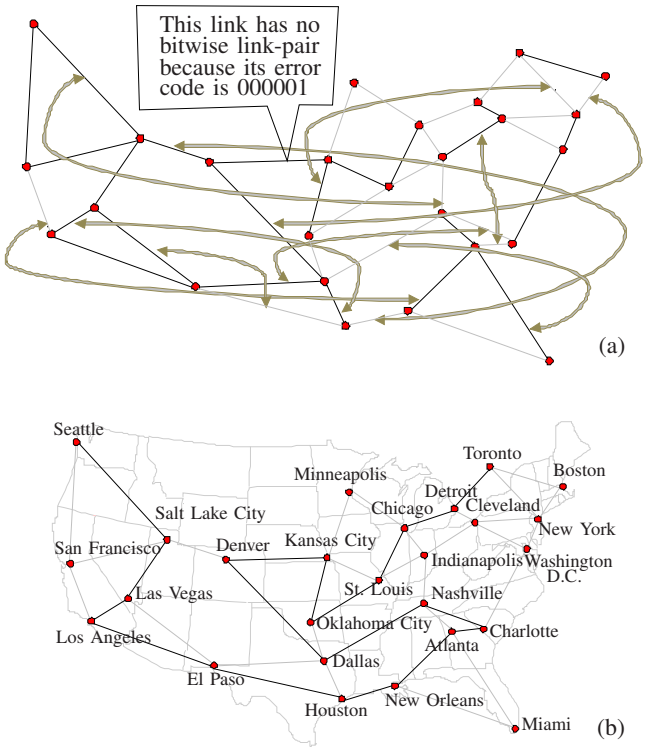


Fig. 7. Link set 6 (a) after random coding (b) after greedy random code swapping for link set 6 in the ACT.

V. Simulation

Extensive simulation on thousands of different random topologies is conducted to verify the proposed algorithm. Specifically, we (1) demonstrate the solution quality of the proposed algorithm by comparing with 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 is 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. 8 shows the quality of m-trail solutions in terms of total cost defined in Eq. 1 versus the granted running time

by taking cost ratio $\gamma = 5$. Each data in Fig. 8 is the mean and 95% confidence interval of 30 experiment results under a specific length of granted running time. 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 attested our expectation. Interestingly, the proposed algorithm has generated even better solutions than that by 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 except for 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.

B. Topology Diversity on M-Trail Solutions

This section demonstrates the impact on m-trail solutions due to topology diversity, for which a huge number of experiments on thousands of randomly generated topologies were conducted. Fig. 9 shows the minimal number of m-trails versus network connectivity 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. To make it statistically meaningful, every data in Fig. 9 is the mean and 95% confidence interval over 20 different topologies each obtained by randomly adding a same number of links on the backbone ring. We observed that the normalized length of alarm code (i.e., the length of alarm code subtracted by $\lceil \log_2 (|E| + 1) \rceil$) dramatically goes down when number of added links and average nodal degree is increased from 50 to 100 in Fig. 9(a) and from 2 to 3 in Fig. 9(b), respectively. The length of alarm code approaches to the lower bound (i.e., $\lceil \log_2 (|E| + 1) \rceil$) when γ is large enough. From Fig. 9(a) and (b), we have also observed that when γ is small, the confidence interval for each data is larger than that in the case with 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. 9(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 alarm code and bandwidth cost by manipulating the cost ratio γ.

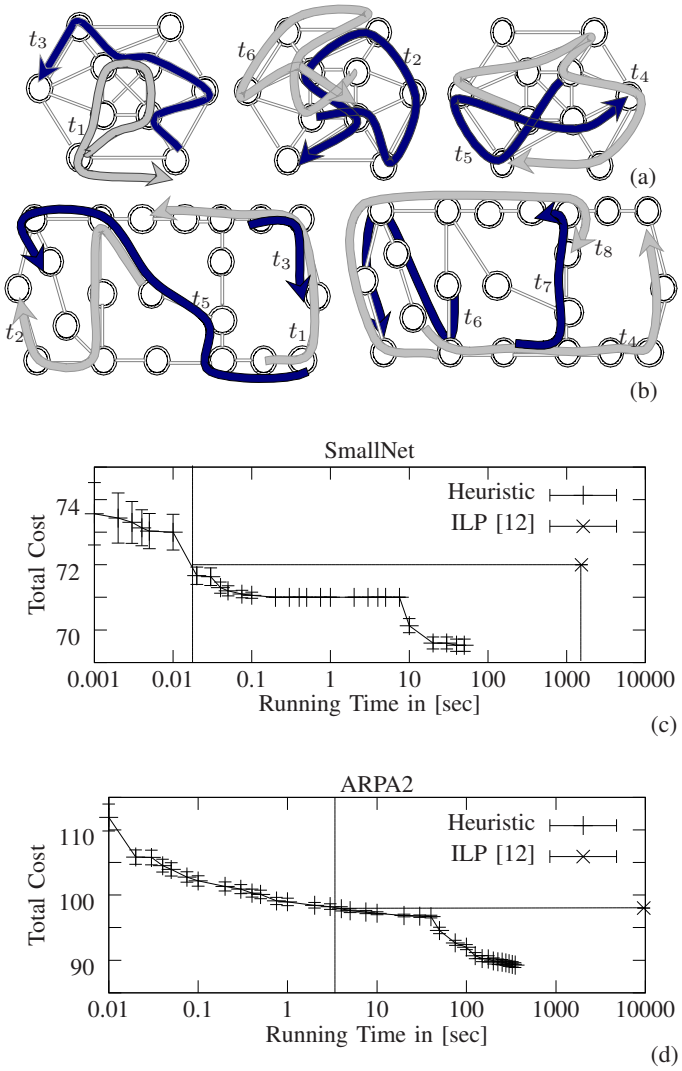


Fig. 8. The best heuristic solutions for SmallNet with cost of 69 and ARPA2 with cost of 87. In comparisons the ILP resulted 72 with 4.17% gap-to-optimality for and SmallNet and 98 with 20.41% for ARPA2.

Fig. 10 shows the m-trail solutions with different numbers of nodes in the network topologies. Fig. 10(a) and (b) shows the length of alarm code versus the number of links. Clearly, when γ = 1000, the length of alarm code is only affected by the number of links when it is small, while quickly converging to $\lceil \log_2(|E| + 1) \rceil$ when the number of added links is increasing, regardless of 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. 10(c) shows the experiment results on the relationship between normalized length of 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.

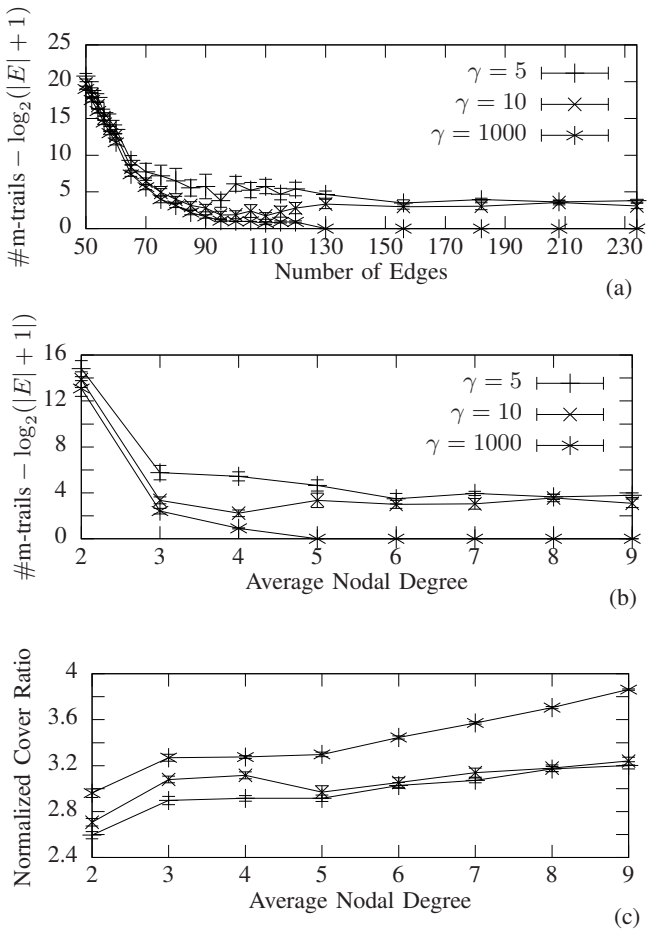


Fig. 9. Simulation on topologies with 50 nodes. (a), (b): normalized length of alarm code versus number of links (or the average nodal degree); and (c): normalized cover ratio versus average nodal degree.

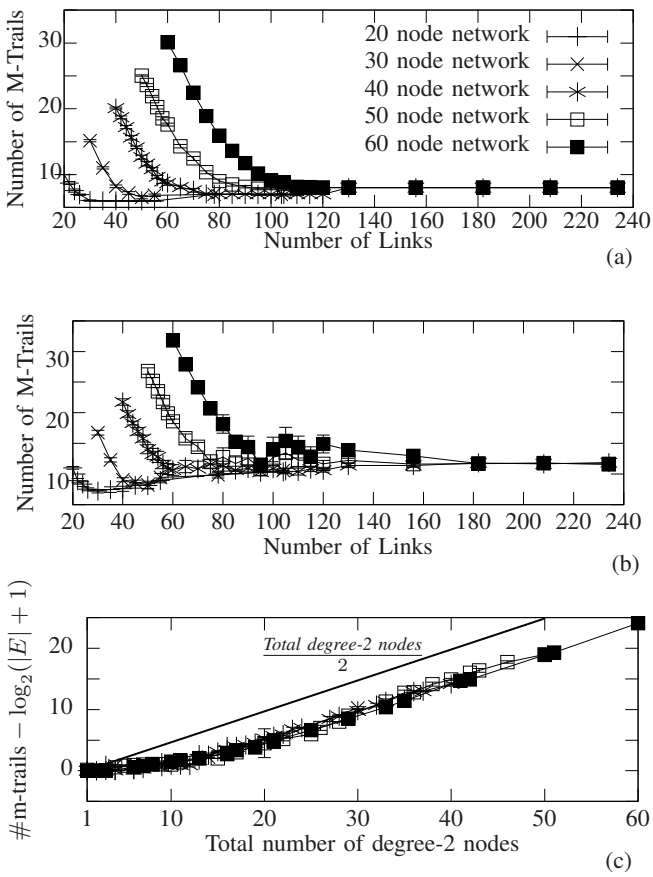


Fig. 10. The length of alarm code versus the number of added links for topologies of 20, 30, 40, 50, and 60 nodes. (a) $\gamma = 1000$ and (b) $\gamma = 5$.

γ is set 1,000. Each data 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 of 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 deterministic effect and 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 alarm code in a general network topology when γ is very large:

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

VI. Conclusive Remarks

As a generalization of all the previously reported all-optical monitoring structures, Monitoring-Trail (m-trail) has been an effective approach for achieving unambiguously failure localization (UFL) under any single failure. The paper investigated the m-trail design problem by firstly obtaining the minimum length of alarm code (or equivalently, the minimum number of m-trails) for ring and fully meshed topologies. To achieve 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 with that by an ILP, where much shorter computation time is required without losing any solution quality.

With the proposed algorithm, the impacts on m-trail solutions due to 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 for achieving UFL is decreased as the network connectivity is increased. The minimum length of alarm code approaches to $\lceil \log_2(|E|+1) \rceil$, 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$\sim$4. (2) When the emphasis on monitoring cost and bandwidth cost is comparable (i.e., γ is small and 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 is increased due to a smaller γ. (3) The minimum length of alarm code can be generally approximated by way of the total number of degree-2 nodes in the topology as shown in Eq. (2).

We expect that the research not only contributes to the failure localization in all-optical networks, but also possibly helps the design of monitoring systems for military surveillance, quantum computing, and production automation, etc. Also, this study is fundamental to the effort of unambiguous localization for multiple simultaneous failures using m-trails, which will be our future study.

REFERENCES

[1] J. Tapolcai, P.-H. Ho, and B. Wu, "Web page on m-trail design: Source codes, 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] ——, "Monitoring Cycle Design for Fast Link Failure Detection in All-Optical Networks," in *IEEE GLOBECOM '07*, 2007, pp. 2315–2319.

[8] C. Li, R. Ramaswami, I. Center, and Y. Heights, "Automatic fault detection, isolation, and recovery in transparentall-optical networks," *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," *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:a new paradigm for fast link failure localization in WDM mesh networks," in *IEEE GLOBECOM '08*, 2008.

[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] 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.

[17] ——, "SRLG Failure Localization in All-Optical Networks Using Monitoring Cycles and Paths," in *IEEE INFOCOM*, 2008, pp. 181–185.

[18] "CPLEX," http://www.ilog.com/products/cplex/.

[19] B. Alspach, J. Bermond, and D. Sotteau, "Decomposition into cycles 1: Hamilton decompositions," *Cycles and Rays*, 1990.

APPENDIX

Lemma 2: A graph can be 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 case (1) and case (2), the cycle and the trails can be formed in linear time with Fleury's algorithm, respectively. ■

Lemma 3: In any graph with $|E_G|$ links, UFL can be achieved with $\lceil \log_2 (|E_G|+1) \rceil$ link sets.

Proof: Let unique alarm codes except $00\ldots0$ be assigned to the $|E_G|$ links one after the other. The alarm code of each link can be converted into an alarm code of $\lceil \log_2 (|E_G|+1) \rceil$ bits, which define $L_1^G, L_2^G, \ldots, L_{\lceil \log_2 (|E_G|+1) \rceil}^G$ link sets such that link l belongs to the i^{th} link set, denoted by L_i^G, if $a_i = 1$ at link l. ■

With Lemma 3 the UFL can be achieved by $b_1 = \lceil \log_2(|E_1|+1) \rceil$ link sets for G_1 denoted by $L_1^{G_1}, L_2^{G_1}, \ldots, L_{b_1}^{G_1}$, and by $b_2 = \lceil \log_2 (|E_2|+1) \rceil$ link sets for each of G_2 and G_3, denoted by $L_1^{G_2}, L_2^{G_2}, \ldots, L_{b_2}^{G_2}$ and $L_1^{G_3}, L_2^{G_3}, \ldots, L_{b_2}^{G_3}$ respectively. Note that $b_1 + 4 \geq 2 \cdot b_2$ because

$$
\begin{aligned}
2 \cdot b_2 - b_1 &= 2\lceil \log_2(|E_2|+1) \rceil - \lceil \log_2(|E_1|+1) \rceil \\
&= 2\lceil \log_2 (2|V|-2+1) \rceil - \lceil \log_2 ((|V|-8)(|V|-1)/2+1) \rceil \\
&\leq 2 \cdot \lceil 1 + \log_2 |V| \rceil - \lceil \log_2 (|V|^2/4) \rceil \\
&\leq 4 + \lceil 2 \log_2 |V| \rceil - \lceil 2 \log_2 |V| \rceil = 4 \quad (3)
\end{aligned}
$$

where $|E_1| = |E| - 4 \cdot (|V|-1) = (|V|-8)(|V|-1)/2$ and $(|V|-8)(|V|-1)/2 + 1 \geq |V|^2/4$ holds for $|V| \geq 4$ based on the quadratic formula. Next, these link sets can be merged and m-trails are shaped based on the following lemma.

Lemma 4: Let $L_j^{G_1}$ and $L_j^{G_2}$ be merged as a single link set for all $j = 1, 2, \ldots, b_2$, denoted as $L_j^{G_{1,2}} = L_j^{G_1} \cup L_j^{G_2}$. $L_j^{G_{1,2}}$ can always be shaped into a valid m-trail t_j by possibly taking the links of G_3 in the m-trail formation.

Proof: Let $\hat{L}_j^{G_3}$ be the set of links added to $L_j^{G_{1,2}}$ forming m-trail t_j, formally $t_j = L_j^{G_{1,2}} \cup \hat{L}_j^{G_3}$. Recall that G_3 consists of two disjoint Hamiltonian cycles. Thus, the links in one of the Hamiltonian cycles can be added to $\hat{L}_j^{G_3}$ such that the connectivity of the link set is guaranteed.

Meanwhile, by staring at an arbitrary node of the second Hamiltonian cycle, we can check each node one-by-one along the cycle. If a node with an odd nodal degree is encountered, the link of next hop at the node is added to $\hat{L}_j^{G_3}$ and otherwise disregarded. This step is repeated until the number of odd-degree nodes is reduced to two and an Euler path can traverse all the links in the subgraph of $L_j^{G_{1,2}} \cup \hat{L}_j^{G_3}$. ■

Using the construction of Lemma 4, $L_1^{G_3}, \ldots, L_{b_2}^{G_3}$ can be merged into the rest of the L^{G_1} link sets and by possibly taking links of G_2 they can be shaped into a valid m-trail. Finally, the link sets of L^{G_3} that were not merged into L^{G_1} (no more then four) can be shaped into a valid m-trails by possibly taking links in G_2.