Failure Localization for Shared Risk Link Groups in All-Optical Mesh Networks using Monitoring Trails

János Tapolcai*, Pin-Han Ho†, Lajos Rónyai‡, Péter Babarczi*, and Bin Wu†
* Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics (BME), {tapolcai,babarczi}@tmit.bme.hu
† Dept. of Electrical and Computer Engineering, University of Waterloo, Canada, {pinhan,b7wu}@bbcr.uwaterloo.ca
‡ Computer and Automation Research Institute Hungarian Academy of Sciences (MTA SZTAKI), 2^{nd} Inst. of Mathematics, BME, ronyai@sztaki.hu

***Abstract*—This paper considers the problem of out-of-band failure localization in all-optical mesh networks using bi-directional monitoring trails (m-trails), where every possible link set with up to d arbitrary links is considered as a shared risk link group (SRLG). With the SRLG scenario, the m-trail allocation problem is firstly formulated, which includes the phases of code assignment and m-trail formation. In the first phase, each SRLG is uniquely coded by assigning each link with a non-adaptive $\overline{d}$ separable combinatorial group testing (CGT) code. Then, the second phase manipulates a sophisticated yet efficient m-trail formation process through a novel greedy code swapping (GCS) mechanism, such that any SRLG failure can be unambiguously localized by collecting the alarms of the interrupted m-trails. The algorithm prototype can be found in [1]. Extensive simulation is conducted on hundreds of randomly generated planar topologies to verify the proposed approach in terms of the number of required m-trails and the computational efficiency. Our approach is compared with previously reported counterparts, by which its merits are further demonstrated.**

***Index Terms* — combinatorial group testing, failure localization, shared risk link group.**

I. Introduction

Real-time and instant localization of fiber cut is a critical task for achieving a fast and failure-dependent traffic restoration in a distributed controlled all-optical mesh networks, and has been considered as a very difficult job due to the transparency in the optical domain along with various design requirements [2]–[4]. One of the most challenging issues is that an upstream link failure may generally trigger redundant alarms by the monitors equipped at the downstream nodes. Besides, a failure at the optical layer (such as a fibercut) may trigger alarms in networking as well as other upper protocol layers [5]. It is reported that a single fiber-cut with 16 disrupted wavelengths could lead to hundreds of alarms in the network [3]. This not only increases management cost of the control plane, but also makes the failure localization difficult.

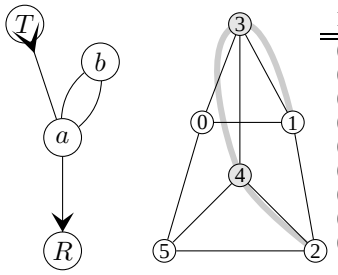
Without loss of generality, the device that monitors the health of a certain part of the network is called a monitor, which generates an alarm if it detects a status change of monitoring results. An alarm is then broadcast in the control plane via a link state protocol, such as Open Shortest Path First (OSPF), so that remote routing entities can localize the failure. To simplify the failure management and operational complexity, it is critical to reduce the number of monitors without sacrificing the accuracy in failure localization.

All-optical out-of-band monitoring via a set of pre-cross-connected supervisory lightpaths for fiber segments has been considered an effective approach to achieve fast failure localization in all-optical backbones. In the past, related studies using various monitoring structures, including monitoring-cycles (m-cycles), m-paths, and m-trails, etc., have been extensively reported [6]–[17]. More detailed comparison and descriptions can be found in [18].

All the previously reported schemes claimed the ability of *unambiguous failure localization* (UFL) for one or multiple failed links, and they aimed at reducing the number of required supervisory wavelength-links, monitoring structures, and/or monitoring locations (MLs), etc. It has been well recognized that with more flexible structures (e.g., m-trails), a better performance can be achieved in monitoring structure allocation, at the expense of higher computation complexity. Bearing this in mind, the paper considers bi-directional m-trails for failure localization of shared risk link groups (SRLGs) with up to d arbitrary links. The m-trail approach is characterized by its flexibility in exploring the network topology diversity. It generalizes all the previously reported counterparts. An m-trail can be a non-simple path/cycle with loop-back switching which allows a node to be traversed by multiple times and a link twice (along both directions).

With bi-directional m-trails, the transmitter and receiver can be allocated at any node pair or co-allocated at a common node along the m-trail. The receiver is equipped with a monitor which alarms in the event of an unexpected and abrupt status change of the corresponding supervisory lightpath. An example is shown in Fig. 1(a), where the transmitter and receiver of the m-trail is denoted by T and R, respectively, and the supervisory lightpath is $T \to a \to b \to a \to R$. Note that the loopback switching at node b makes both directions of the links $a-b$ and $b-a$ traversed by the lightpath. It will not affect the monitoring result by having different connection patterns on a set of links or different locations for the transmitter and receiver, because we only care about whether the supervisory lightpath is disrupted or not.

Fig. 1 shows an example of m-trail solution for localizing



Link	t_1	t_2	t_3	t_4
(3,4)	1	0	1	0
(2,5)	0	0	0	1
(2,4)	1	0	0	1
(1,3)	1	0	0	0
(1,2)	0	1	0	1
(0,3)	0	1	1	0
(0,5)	0	0	1	1
(4,5)	0	0	1	0
(0,1)	0	1	0	0

(a) M-Trail (b) Topology and t_1 (c) Alarm code table

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

any single link failure, where an *alarm code table* (ACT) is shown in Fig. 1(c). The ACT keeps the alarm code of each link (e.g., link $(3, 4)$ is assigned an alarm code 1010), which further defines how the four m-trails (i.e., t_1, t_2, t_3, and t_4) should be routed in the topology to achieve UFL. Here, t_j has to traverse through all the links with the jth bit of the alarm code as "1" while avoiding to take any link with the jth bit of its alarm code as "0". By reading the status of the four m-trails, any link failure can be unambiguously localized. For example, the darkness of t_1 and t_3 depicts the failure of link $(3, 4)$.

Although localization of single link failure in all-optical mesh networks has been extensively studied, to the best of our knowledge, only [15]–[17] have investigated the failure scenarios of multiple links. However, all the schemes developed in [15]–[17] take topology connectivity as an important constraint, which have imposed an inherent performance barrier to those schemes. For example, the probing tree construction by [15] is valid only if the network topology is sufficiently densely meshed; while the m-paths and m-cycles employed in [16], [17] are obtained using a least-cost routing algorithm. The above studies obviously leave some space to improve.

In this paper, we consider the problem of bi-directional m-trail allocation for achieving UFL of SRLGs with up to d arbitrary links. To ensure code uniqueness of each SRLG, $\overline{d}$ seperable codes are generated and assigned to each link via a suite of state-of-the-art non-adaptive combinatorial group testing (CGT) code constructions. With a unique code on each SRLG, a sophisticated yet efficient mechanism, called *greedy code swapping* (GCS), is developed for efficiently exploring the design solution space.

The proposed approach is examined via extensive simulation over hundreds of randomly generated topologies. The performance metrics of interest in the study are the number of m-trails and the computational efficiency, by which a comparison is made with a naive and widely employed scheme based on sequentially allocating monitoring structures to distiguish each pair of SRLGs [16], [19], [20]. Extensive simulation is conducted to verify the proposed approach and compare with previously reported counterparts, where significant performance improvement is witnessed.

The rest of the paper is organized as follows. Section II provides a survey on the related studies. Section III presents the problem formulation. Section VI introduces the proposed approach for the failure localization problem. Section V shows the simulation results. Section IV concludes the paper.

II. Related Work

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

(R1): Every SRLG should be uniquely coded.

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

In addition to (R1) and (R2), there could be some other constraints due to specific user design premises, such as the length limitation due to the deployment of optical generators/retransmitters, the locations of monitoring nodes [16], [21], and use of working lightpaths (i.e., live connections) for failure state correlation [16], [22]. Without loss of generality, this study focuses on (R1) and (R2), which are the fundamental for an m-trail UFL solution. Nonetheless, we claim that our approach can easily tackle any additional requirement imposed upon the proposed m-trail allocation problem.

An integer linear program can be developed that satisfies both (R1) and (R2) in a single step [9], [14], [23], which is unfortunately subject to intolerably long computational time even in very small topologies. Thus people have turned to the design of heuristics in solving the problem. The previously reported solutions can be divided into two categories according to their design principles. The first one manipulates an accumulation mechanism such that (R2) is ensured at the beginning, while the goal of the heuristics is to satisfy (R1) [15]–[17]. In the second design category, (R1) is intrinsically ensured at the beginning while leaving (R2) as a goal [24].

In [15], with the help of combinatorial group testing (CGT) code constructions the authors conducted an indepth theoretical bound analysis on the minimum number of permissible probes required for localizing a failed SRLG with up to d arbitrary links, in which each link is assigned with multiple codes in a graph with at least $d + 1$ disjoint spanning trees. Therefore, the construction in [15] can only be applied to very densely meshed topologies. For example, the network has to be as densely meshed as $(2d + 2)$-connected in order to accommodate $d + 1$ disjoint spanning trees, which results in $(d + 1) \cdot J$ bits assigned to each link for achieving UFL of SRLGs with up to d links, where J is the CGT code length. Obviously, such a method by assigning each link $d + 1$ CGT codes can well fit into theoretical analysis, but it can hardly be applied in most practical scenarios.

The studies in [16], [17] set their goal in minimizing the number of monitoring locations (MLs). For example, to localize failure of SRLG with up to 2 links (i.e., $d = 2$), all the 3- and 4-connected subgraphs should be identified, and almost each subgraph needs an ML at an arbitrarily chosen node in the subgraph. With each ML determined, graph transformation is performed such that the MLs are merged into a supernode (denoted as m), and cycles are cumulatively added into the transformed graph one by one via Suurballe's [25] algorithm.

To distinguish two SRLGs w_1 and w_2, a cycle must be disjoint from w_2 while passing m and l, where l is a link randomly selected from $w_1 \setminus w_2$. This leads to $O(|SRLG|^d)$ of cycles that are necessary to distinguish all the SRLGs, where $|SRLG|$ is the number of SRLGs considered in the network. Thus, the worst time complexity is $O(|SRLG|^d \cdot |V|^2)$, where $|V|$ is the number of nodes in network, and the term $O(|V|^2)$ corresponds to the complexity of Suurballe's algorithm. The computation complexity becomes $O(|E|^{2d} \cdot |V|^2)$ if every single-link and dual-link failure should be localized, where $|E|$ is the number of links.

The approach taken in [24] is the first study following the second design principle, where the code uniqueness of each link (as defined in (R1)) is first guaranteed, while an algorithm was given for the formation of each monitoring structure (known as m-trail). A superb performance was witnessed in [24] by employing random code assignment (RCA) and random code swapping (RCS) for localizing any link failure. However, the study in [24] was only for a single-link failure.

This study follows the second design category in order to take advantage of the extremely flexible structure of bi-directional m-trails in solving the problem.

III. Problem Formulation

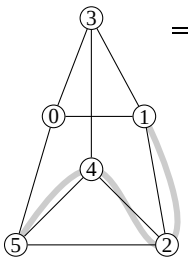
The target of the m-trail design is to allocate a set of m-trails for localizing failure of an SRLG with up to arbitrary d links according to a give cosst function. With the second design principle, the proposed approach divides the m-trail allocation problem into two sequential tasks: code assignment for achieving (R1) followed by m-trail formation that ensures (R2).

A. Cost function

The objective of m-trail allocation is to minimize the weighted sum of *monitoring cost* and *bandwidth cost*. The monitoring cost includes the hardware cost and the control complexity [14], [24]. Let the *length* of an m-trail be the number of hops it traverses, and the *cover length* be the length sum of all the m-trails in the solution. The total cost function is expressed as:

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

The *cost ratio* β specifies the relative importance of monitoring cost and bandwidth cost. In general, the monitoring cost concerns not only the monitors' expense but also the efforts of network control and fault management, while wavelengths are getting cheaper. Thus, it is a usual case that β is chosen much larger than 1. In this paper, β is taken as 1000 to reflect this fact, where the effect of bandwidth cost is negligible in routing the m-trails.



Link	t_1	t_2	t_3	t_4	t_5	t_6	t_7
(3,4)	0	1	0	0	0	1	0
(2,5)	0	0	1	0	0	1	1
(2,4)	1	0	0	0	0	1	0
(1,3)	0	1	0	1	0	0	0
(1,2)	1	0	0	1	0	0	1
(0,3)	0	0	1	1	0	0	0
(0,5)	0	0	1	0	1	0	0
(4,5)	1	0	0	0	1	0	0
(0,1)	0	1	0	0	1	0	1

(a) Topology and t_1 (b) Alarm code table

Fig. 2. Alarm codes for dual-link failure localization.

B. Code Uniqueness

In the single-link failure scenario, maintaining code uniqueness of each link is straightforward since the SRLGs are single-links and are independent from each other. In this case, code uniqueness of each SRLG can be achieved by keeping a Hamming distance as small as 1 among different codewords. Let $[a_1^l, a_2^l, \ldots\ldots a_J^l]$ denote an alarm code of J bits for link l, where the binary bit a_j^l is 1 if the j-th m-trail traverses through l and "0" otherwise [24]. The above solution is no longer valid in case all the possible SRLGs with up to d arbitrary links are considered. This can be clearly demonstrated by the example in Fig. 1(c): The dual-link failure on (1,3) and (2,5) results in an alarm code 1001, which has been assigned to link (2,4), causing a collision with the code assigned to $(2, 4)$.

In our study, the alarm code of a multi-link SRLG is the *bitwise OR* of the alarm codes of all links in the SRLG. This corresponds to the fact that a monitor alarms if the corresponding m-trail traverses through any link in the SRLG that is hit by the failure event. Thus, the code uniqueness in the considered scenario can be achieved iff the *bitwise OR* of the codes of links in an SRLG is distinguished from any other possible code existing in the network. As shown in Fig. 2, the *bitwise OR* of any pair of link codes has to be network-wide unique for the corresponding m-trails to achieve UFL for any single- and dual-link SRLG.

In this study, non-adaptive combinatorial group testing (CGT) codes are adopted to ensure the code uniqueness of each SRLG. The primary goal of a CGT construction is to identify up to d defective items among a given set through as few tests as possible. In this case, the set of items are the network links, the defective items are the failed links, and the tests are by way of allocating a set of m-trees in the network. Thus, we need to identify a proper non-adaptive CGT construction that can ensure the uniqueness of the bitwise OR of up to d codes, and the codes can be randomly assigned to the network links. This corresponds to the requirement that all possible multi-link failures with up to d links can be unambiguously localized provided with a successful formation of a set of m-trees. Specifically, the constructions by [26], [27] is employed in the study to generate CGT codes with $\overline{d}$-separability for each link, such that any bitwise OR of up to arbitrary d codes is distinct from each other.

Note that the idea of using non-adaptive CGT codes for multi-link failure localization has been explored in [15];

Nonetheless, the proposed approach attempts to achieve the best informatic efficiency of the assigned CGT codes via a novel code swapping mechanis, rather than statically assigning the codes for each spanning tree [15] which certainly results in a vast amount of tests (or unnecessarily long alarm codes).

C. M-trail Formation

In a nutshell, the task of m-trail formation is to allocate a set of m-trails with the minimum cost in Eq. (1), such that the j-th m-trail, $\forall j$ s.t. $1 \leq j \leq J$, is routed through every link l with $a_j^l = 1$, while disjoint from any link l with $a_j^l = 0$. This constraint on routing the m-trails corresponding to each bit position alarm code is called *coverage constraint*, which imposes very high complexity in m-trail formation especially when J is large. Let L_j and $\overline{L}_j$ denote the link set containing link l with $a_j^l = 1$ and $a_j^l = 0$, respectively. The number of m-trails required to cover all the links in L_j depends on the number of isolated fragments in L_j. On the other hand, the cover length of the m-trail solution is determined by the total number of "1" in the code of each link. Initially, since each link is randomly assigned a CGT code of length J, an m-trail solution under the casual code assignment may lead to bad solution quality due to a lot of fragments (each corresponding to an m-tree) at every bit position, which result in a very large cost to Eq. (1).

IV. Proposed Approach for M-Trail Allocation

The proposed m-tree allocation method follows the second design principle, where CGT codes generated by [26], [27] are assigned to each link to ensure (R1). After the random code assignment, (R2) is pursued by way of greedy code swapping (GCS). Although seemingly similar to that in [24], the proposed approach is much different in both stages of code generation and code swapping.

Fig. 3 is a flowchart that summarizes the proposed approach. In Step (1), the CGT code construction `GEN_CGT` generates a number of k $\overline{d}$-separable codes of a length J bits, denoted as CGT. The input parameter $|E|$ ensures that the code length J is the smallest such that $k \geq |E|$. Note that the property of $\overline{d}$-separability ensures uniqueness of the *bitwise OR* of up to d codes in $\underline{CGT}$, which is required in (R1). In Step (2), an alarm code table is formed by randomly selecting $|E|$ out of k codes from $\underline{CGT}$, which are further assigned to all the links. The group of $|E|$ codes taken by the links is denoted as $\underline{ACT}$, while the group of rest $k - |E|$ unassigned codes is denoted as $\underline{UCT}$, where $\underline{UCT} = \underline{CGT} \setminus \underline{ACT}$.

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

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

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

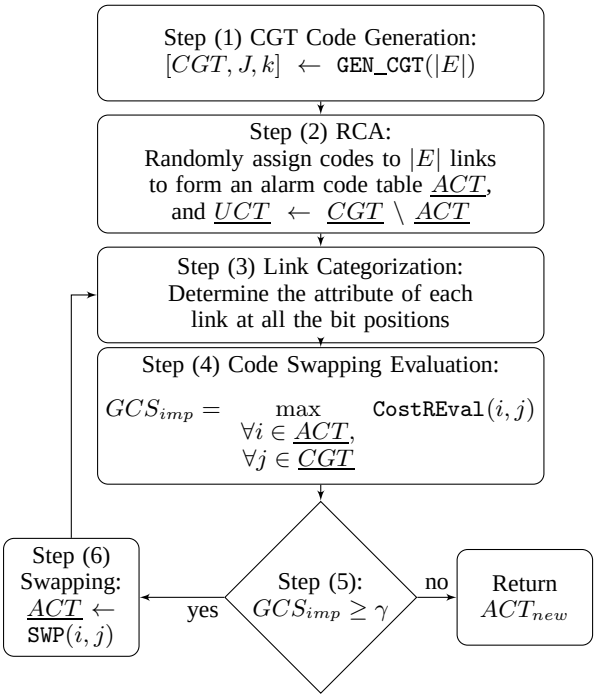


Fig. 3. The flowchart for the proposed heuristic algorithm.

A. Greedy Code Swapping (GCS)

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

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

The flowchart of the proposed GCS is given in Fig. 4, which provides all details of Step (4) in Fig. 3. At the beginning,

the program picks up a code pair C_e and C_x as shown in Step (4.1), where C_e is a code assigned to link e while C_x is randomly selected from $\underline{CGT}$, respectively. The cost reduction evaluation for a single swapping should be iterated on each bit position (or, each link set) affected by the swapping. The i-th bit position (or link set), L_i, is not affected by the swapping of C_e and C_x if the two codes have a common i-th bit, i.e., $C_{e,i} = C_{x,i}$. If the swapping of C_e and C_x has an affection on the i-th link set, the heuristic goes to either Step (4.5) or (4.6), depending on whether $C_x \in \underline{ACT}$ or $C_x \in \underline{UCT}$, which is checked in Step (4.4). In the case $C_x \in \underline{UCT}$, the function `addBit`(e, i) is called if $C_{x,i} = 1$ and $C_{e,i} = 0$; otherwise `removeBit`(i, e) is called if $C_{x,i} = 0$ and $C_{e,i} = 1$. In the former case the i-th bit is flipped from "0" to "1", hence a link is added to L_i; while in the latter, the i-th bit is changed from "1" to "0", where a link is removed from L_i. If $C_x \in \underline{ACT}$, let C_x be currently assigned to link f. The function `add&removeBit`(i, e, f) is called if $C_{x,i} = 1$ and $C_{e,i} = 0$; otherwise the function `add&removeBit`(i, f, e) is called (i.e., $C_{x,i} = 0$ and $C_{e,i} = 1$).

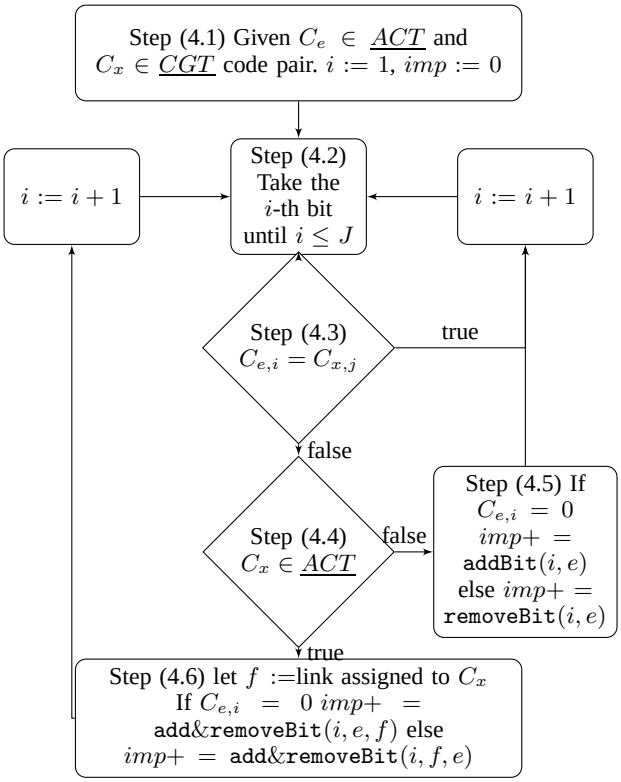


Fig. 4. Cost reduction evaluation for each code swapping.

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

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

Isolated link is a link not connected to any other link of the link set. Identifying these links is simple, since their both terminating nodes have degree 1. An example is given as (x, n) in Fig. 5.

Leaf link is a link with exactly one of its terminating nodes of nodal degree 1, as shown in link (c, r) and (u, z), etc., in Fig. 5.

Bridge link has both terminating nodes with a nodal degree larger than 1. Moreover, if the link is erased, then the component falls apart into two sub-components. To identify a bridge link, every 2-connected component must be identified first, which can be done in $O(|E|^2)$ time. For these links both terminating nodes of the link must belong to different 2-connected components. An example is given as (c, e) in Fig. 5.

Detour link is a part of a component,and removal of it does not tear the component apart. For these links both terminating nodes of the link must belong to the same 2-connected component. An example is given as (o, u) in Fig. 5.

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

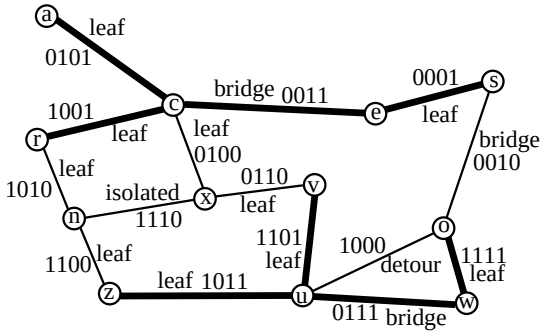


Fig. 5. An example on link attribute categorization for fast cost reduction evaluation on code swapping.

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

TABLE I
TABLE LOOKUP OF ADDBIT(i, e)

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

3) `removeBit`(i, e)*:* returns the cost reduction in case $C_{e,i} = 1$ and $C_{x,i} = 0$. Because the i-th bit is changed

from "1" to "0", the cover length is decreased by 1, while the number of m-trees should be updated according to the attribute of e with respect to L_i. This is summarized in Table II.

TABLE II
TABLE LOOKUP OF REMOVEBIT(i, e)

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

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

TABLE III
TABLE LOOKUP OF ADD&REMOVEBIT(i, e, f)

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

In summary, the proposed GCS swaps a code pair with the steepest cost reduction larger than a threshold γ based on the proposed link attribute categorization and table lookup process, which greedily approaches to better performance according to Eq. (1). With GCS, very high computational efficiency can be achieved thanks to the constant time complexity in evaluating each code pair, which will be detailed in the next subsection. The prototype of the proposed algorithm can be found in [1].

B. Computational Complexity Analysis

The cost reduction evaluation is performed in each code swapping, which dominates the computational complexity of the heuristic algorithm. The following lemma describes the computational complexity of the cost reduction evaluation process for a single code swapping.

Lemma 1: The computational complexity of a cost reduction evaluation process for a single swapping is $O(|E|^2 \log |E|)$.

Proof: The proof of *Lemma 1* is completed via verifying the following three claims.

Claim 1: The complexity of `CostREval`(i, j) is $O(1)$.

Claim 2: The complexity of Step (2) is $O(|E| \log |E|)$.

Claim 3: The complexity of Step (3) is $O(|E|^2 \log |E|)$.

The detailed argument is relegated to Appendix. ■

It is clear that the number of isolated components (or m-trails) in the initial random code assignment for each bit position cannot be more than $|V|/2$. This is because each isolated component consists of at least a single edge and two nodes, where $|V|$ is the number of nodes in the network. After each loop (defined in Steps (3), (4), (5), and (6) of Fig. 3), at least one m-trail is determined and erased from the link set; thus, the maximum number of code swappings should be upper bounded by $\frac{|V|}{2} \cdot J$, where J is the code length (in bits). Note that J is in the order of $O(d \cdot \log |E|)$ according to the CGT construction. By considering the complexity of each code swapping as $O(|E|^2 \log |E|)$, the overall worst case complexity in the proposed method is $O(|V||E|^2 \cdot d \cdot \log^2 |E|)$. Compared with the scheme in [16], [17] with a complexity of $O(|E|^{2d} \cdot |V|^2)$, the proposed approach can achieve much better efficiency.

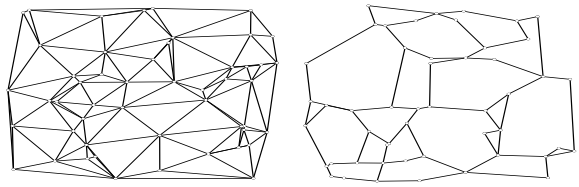
V. SIMULATION

Simulations on hundreds of randomly generated planar 2-connected network topologies were conducted. The network topologies were generated with `lgf_gen`, a random graph generator of LEMON [28], which randomly generates realistic planar 2-connected networks. The networks are classified according to the girth of the graph, denoted by g, which is the length of a shortest cycle contained in the graph. Clearly, a smaller value of g yields a more densely meshed topology. Fig. 6(a) and (b) give two example network topologies, and (c) the statistics of the randomly generated topologies. It is found that the average nodal degree is 3.0 for dense networks ($g = 7$) and 4.0 for sparse networks ($g = 4$).

In the simulation, the proposed scheme is denoted as GCS^1, GCS^2 and GCS^3 for failure localization of SRLGs with up to 1, 2, and 3 links, respectively, where the CGT codes based on d-separable constructions with $d = 1$, $d = 2$, and $d = 3$ are employed. CA^1 and CA^2 corresponds to the method that each bi-directional m-trail is allocated one after the other to distinguish each pair of SRLGs using any Dijkstra's algorithm based scheme, such that UFL for single-link SRLGs and for both single- and dual-link SRLGs can be achieved, respectively. The method is generic and has been considered in a number of previously reported studies [16], [19], [20]. We have also implemented the construction in [15] that used disjoint spanning trees, denoted as $DSTC$. The construction provides an upper bound for $(d + 1)$-connected

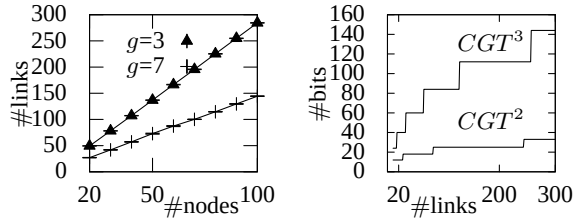
topologies, but is invalid for topologies with any node of a smaller nodal degree than $(d+1)$. The upper bound is given by $(d+1)\dots J$, where J is the length of the CGT codes employed.

Fig. 6(b) shows the lengths of CGT codes (i.e., J) versus the number of links $|E|$ of the corresponding topology by the CGT code generator `GEN_CGT` in [26], where the scenarios with $d=2$ are presented. It is intuitive that when SRLGs with more links are considered, longer CGT codes are required for each link.



(a) An example of 50 node maximum planar network (girth parameter $g=3$).

(b) An example of 50 node network with girth parameter $g=7$.



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

(d) The length of CGT code (in bits) versus the number of links as an input to the CGT code generator `GEN_CGT` in [26]

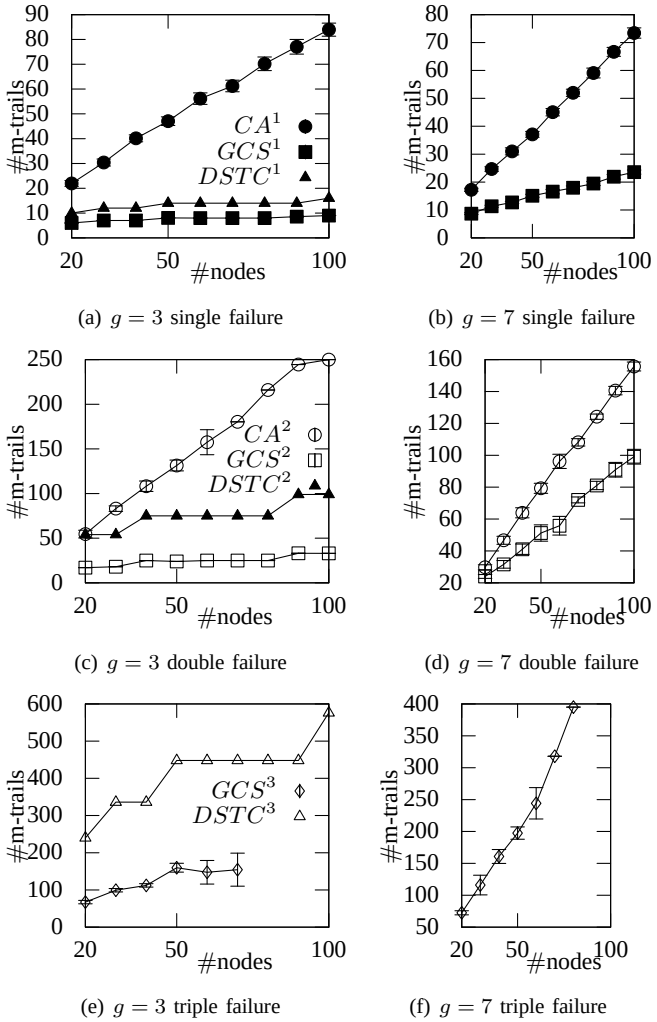
Fig. 6. Statistics on the input data.

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

A. Number of M-Trails versus Network Size

The performance in terms of the minimum number of m-trails is first investigated, and the results are shown in Fig. 7. First, we find that the number of m-trails increases when the network size grows, which is observed in all the cases. It clearly shows that the proposed approach achieve much better scalability, where GCS^1, CA^1, and CA^2 have achieved far worse performance than that by GCS^1 and GCS^2, respectively, in both types of network topologies.

The superior performance of the proposed approach in minimizing the number of m-trails can be explained in two folds. First, the m-trails have the most flexible routing structure that can fully explore the solution space. This serves as a critical factor in overcome the vicious effect of topology diversity. It can be attested that our scheme has less advantage against the other two counterparts when network is sparsely connected (i.e., $g=7$), because there are less alternatives in allocating the m-trails. Second, because CA^1 and CA^2 have each monitoring lightpath sequentially allocated into the network using an shortest path routing algorithm, it lacks intelligence in exploring the design space and network topology diversity. Third, $DSTC$ [15] tries to ensure the code uniqueness of each SRLG by useing $d+1$ disjoint spanning trees, which is strongly limited by the topology connectivity. It is clearly shown that the construction can only yield valid solution in very densely meshed topologies, while failed in most of the sparse topologies considered in the simulation.



(a) $g=3$ single failure

(b) $g=7$ single failure

(c) $g=3$ double failure

(d) $g=7$ double failure

(e) $g=3$ triple failure

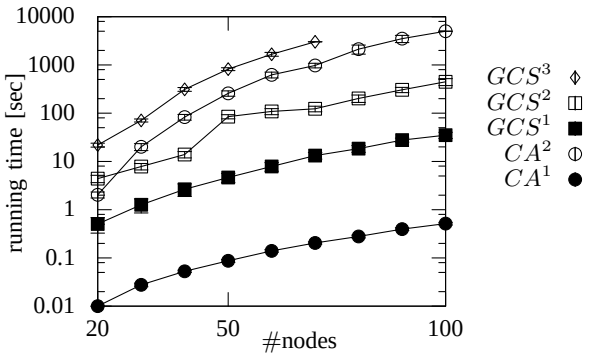
(f) $g=7$ triple failure

Fig. 7. The number of m-trees versus the number of nodes.

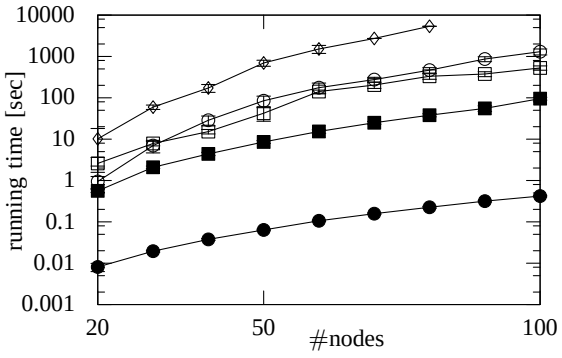
B. Running Time

Fig. 8 shows the running time for obtaining the data in Fig. 7.It is observed that the proposed approach achieves much better computational efficiency, although CA^2 can achieve performance closer to GCS^2 for sparser network topologies. Also, GCS^3 takes much longer time than GCS^2 and any other case due to a much longer CGT code with $d = 3$, as shown in Fig. 6(b). Recall that the cost reduction evaluation in each code swapping has to go through all the link sets that are affected by the code swapping. Thus, the longer CGT code of each link yields an immediate increase of running time in obtaining an m-trail solution.

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



(a) Dense networks



(b) Sparse networks

Fig. 8. The running time versus the number of nodes.

VI. Concluding remarks

The paper has investigated the problem of SRLG failure localization in all-optical mesh networks using bi-directional m-trails, which aims to achieving unambiguous localization of a failure event that affects up to d arbitrary links. We firstly defined the code uniqueness in the SRLG failure localization scenario, and introduced a novel greedy code swapping (GCS) mechanism that can well handle the connectivity requirement in the m-trail formation. Simulation was conducted over hundreds of randomly generated planar topologies to examine the proposed approach when the number of links contained in each SRLG is 1, 2, and 3, respectively. Comparison was made with a couple of previous arts in terms of the minimum number of required m-trails and running time. The simulation results verified the scalability of the proposed heuristic algorithm as the network size increases, and demonstrated a significant performance gain over its counterparts in all possible scenarios investigated in the study. We conclude that the superiority of the proposed approach is achieved by an effective incorporation with CGT codes in ensuring code uniqueness, the employment of the bi-directional m-trails, as well as an intelligent GCS mechanism that can make the best use of the extremely flexible and general structure of m-trails.

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] I. Tomkos, "Dynamically Reconfigurable Transparent Optical Networking Based on Cross-Layer Optimization," in *ICTON '07*, vol. 1, 2007, pp. 327–327.

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

[4] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE network*, vol. 14, no. 6, pp. 16–23, 2000.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[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] 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.
[19] H. Zeng, C. Huang, and A. Vukovic, "A novel fault detection and localization scheme for mesh all-optical networks based on monitoring-cycles," *Photonic Communication Networking*, pp. 277–286, 2006.
[20] 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*, 2002, pp. 2676–2680.
[21] B. Wu, P.-H. Ho, J. Tapolcai, and X. Jiang, "A novel framework of fast and unambiguous link failure localization via monitoring trails," in *Proc. IEEE INFOCOM WIP*, San Diego, 2010.
[22] 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.
[23] B. Wu, P. H. Ho, J. Tapolcai, and P. Babarczi, "Optimal Allocation of Monitoring Trails for Fast SRLG Failure Localization in All-Optical Networks," in *IEEE Globecom*, 2010.
[24] 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.
[25] J. W. Suurballe, "Disjoint Paths in a Network," *Networks*, vol. 4, pp. 125–145, 1974.
[26] D. Eppstein, M. Goodrich, and D. Hirschberg, "Improved combinatorial group testing for real-world problem sizes," in *Workshop on Algorithms And Data Structures (WADS)*. Waterloo, ON, Canada: Springer, Aug. 2005, pp. 86–98.
[27] F. K. Hwang and V. T. Sós, "Non-adaptive hypergeometric group testing," *Studia Sci. Math. Hungar*, vol. 22, pp. 257–263, 1987.
[28] "Lemon: A c++ library for efficient modeling and optimization in networks," http://lemon.cs.elte.hu.
[29] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

VII. Appendix

A. Proof of Lemma 1

Proof of Claim 1: In the table lookup process for each code swapping, it can be intuitively verified that every entity in the table can be performed in constant time.

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

For link set L_j, we need to determine the relationship between any node and a bridge link of $a_j = 1$, which can be done in constant time. For example in Fig. 5(a), (c, e) is a bridge link of L_j, and removal of it separates L_j into two isolated components that form two m-walks with nodes $\{a, c, r\}$ and $\{e, s\}$, respectively. Such a function can be implemented by storing the *reach* and *leave* order of each node in the DFS algorithm. As shown in Fig. 9 as an example, the reach order of the DFS is written on the top of nodes, while the leave order is below the nodes. Let I_R and I_R denote the largest reach and the smallest leave order indices of the bridge. Every node with a reach and leave index at least I_R and at most I_L belongs to one side of bridge. As exemplified in Fig. 9, we have $I_R = 2$ and $I_L = 8$, thus $\{a, r, c\}$ are in one sub-component. ■

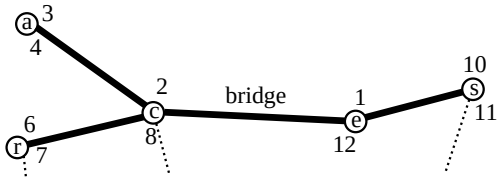


Fig. 9. Leave and reach order of the DFS algorithm. The links of L_j are drawn with thick lines.

Proof of Claim 2: Clearly, a DFS function for detecting the components in each link set is in $O(|E|)$ time complexity. Since each bridge link connects to two 2-connected components, to identify a bridge link, we must identify the corresponding two 2-connected components first, which can be done in $O(|E|)$ time complexity [29]. Also, such a check needs to go through each bit position, which multiplies the complexity by a factor of $\log |E|$. Thus, claim 2 is proved. ■

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

B. SRLG failure localization is NP-complete

Definition 1: M-tree allocation problem (MTA):

Instance: a network $G(V, E)$, a $\mathcal{F}$ set of SRLGs, a positive integer $b < |E|$ representing a limit on the length of the alarm code.

Question: Is it possible to deploy $\leq b$ m-trees in order to unambiguously localize all SRLGs in $\mathcal{F}$, i.e., for any pair of SRLGs $e, f \in \mathcal{F}$, (1) there is an m-tree which passes through e but not f or vice versa, and (2) all SRLGs are passed by at least one m-tree?

Definition 2: Hitting set problem:

Instance: Collection C of subsets of a finite set S, positive integer $K < |S|$.

Question: Is there a subset $S' \subseteq S$ with $|S'| \leq K$ such that S' contains at least one element from each subset in C?

A Karp-reduction is shown, in the case the graph is not connected and not all single-link SRLGs are localized.

Lemma 2: The MTA problem is **NP**-complete.

Proof: The MTA problem is in **NP**, as verify whether a candidate solution unambiguously localize all SRLGs with $\leq b$ m-trees or not can be done in polynomial time.

Assume we are given an instance of the hitting set problem, that is, a finite set S with n elements $s_1, s_2, \ldots, s_n$, and a collection of subsets $C_1, C_2, \ldots, C_r$.

The polynomial time transformation is given as follows. We construct a graph with $n + r$ isolated edges, where isolated e_i is assigned for each $s_i, i = 1, \ldots, n$ in the set, and isolated edge f_j is assigned for each subset $C_j, j = 1, \ldots, r$. Note that the graph consists of isolated edges, thus all m-trees consists of a single edge. An SRLG is defined for all $C_j = \{s_{j_1}, s_{j_2}, \ldots, s_{j_k}\}$ as $\text{SRLG}_j = \{e_{j_1}, e_{j_2}, \ldots, e_{j_k}, f_j\}$,

and SRLG$_{r+j} = \{f_j\}$. In the rest of the proof we show that $2r$ SRLGs can be unambiguously localized with $\leq b = K + r$ m-trees, if and only if ($\Leftrightarrow$) the hitting set problem is solvable with $\leq K$ elements.

($\Leftarrow$) Suppose there exists a solution for the hitting set problem with K elements. In this case, MAP problem has a solution with $b = K + r$ m-trees. In the m-tree solution, all edges f_j for $j = 1, \ldots, r$ are covered by a single m-tree. Besides, the e_{j_k} edges corresponding to element s_{j_k} in the hitting set are also covered by a single m-tree. In this case, the SRLGs are unambiguously localized, because (2) all SRLGs are covered with at least one m-tree. And (1) the m-tree on f_i passes through SRLG$_i$ and SRLG$_{i+r}$, but none of the other SRLGs. The two SRLGs can be distinguished if there exists an m-tree, which passes through SRLG$_{i+r}$, but not SRLG$_i$. Such m-tree exists, as SRLG$_{i+r}$ is passed by another m-tree on single edge e_{j_k}, which corresponds to the s_{j_k} element in the hitting set.

($\Rightarrow$) Finally, we show how to convert an m-tree solution with $b = K + r$ m-trees into a K element solution of the corresponding hitting set problem. The first observation is that the MTA solution has r m-trees with single links of f_j for $j = 1, \ldots, r$ for unambiguously localize SRLG$_i, i = r+1, \ldots, 2r$. The second observation is that at least one edge e_i from each SRLG$_j, j = 1, \ldots, r$ are covered by an m-tree in order to distinguish the failure of SRLG$_i$ and SRLG$_{i+r}$. Since edges e_i for $i = 1, \ldots, n$ are passed by $K = b - r$ single edge m-trees and, the K elements in S corresponding to these m-trees forms a hitting set on C_j. Thus, the MTA is **NP**-complete. ■